This electronic thesis or dissertation has been downloaded from the King's Research Portal at https://kclpure.kcl.ac.uk/portal/

KING'S College LONDON

SELF-LOCALIZATION AND ENVIRONMENT BUILDING METHODS FOR SMALL NONHOLONOMIC MANOEUVRABLE TWO-WHEEL MOBILE ROBOTS

Georgiou, Evangelos

Awarding institution:
King's College London

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without proper acknowledgement.

END USER LICENCE AGREEMENT

Unless another licence is stated on the immediately following page this work is licensed

under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International

licence. https://creativecommons.org/licenses/by-nc-nd/4.0/

You are free to copy, distribute and transmit the work

Under the following conditions:

- Attribution: You must attribute the work in the manner specified by the author (but not in any way that suggests that they endorse you or your use of the work).
- Non Commercial: You may not use this work for commercial purposes.
- No Derivative Works - You may not alter, transform, or build upon this work.

Any of these conditions can be waived if you receive permission from the author. Your fair dealings and other rights are in no way affected by the above.

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 04. May. 2023

SELF-LOCALIZATION AND ENVIRONMENT BUILDING METHODS FOR SMALL NONHOLONOMIC MANOEUVRABLE TWO-WHEEL MOBILE ROBOTS

Evangelos Georgiou
King's College London

The Center for Robotics Research (CoRe)
Department of Informatics
King's College London

Submitted 21st January 2014

This thesis is submitted to the Department of Informatics, King's College London, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Abstract

The thesis presents a kinematic and dynamic model of the mobile robot platform derived by Lagrange D'Alembert methodologies and system control using a closed-loop PD controller. Innovative research in self-localization is presented in this thesis with the use of a double compass configuration that exploits a fusion of relative and absolute localization methods to achieve an analytical solution to position.

In order to validate this novel double compass self-localization method, an optimized method was proposed in the form of an overhead computer system and a two-wheel manoeuvrable nonholonomic mobile robot was developed to facilitate research in self-localization methods with shaft encoders, accelerometers, magnetometers, and gyroscopes. The computer system was used to improve the performance of track non-natural markers on the mobile robot. A novel pseudo random algorithm with a gradient policy, inspired by the skip-list method, was delivered to significantly improve the image scanning performance to find non-natural markers. The validation, analysing the data collected from double compass configuration compared to visual tracking data was carried out using a non-parametric single-sample statistical analysis using the Kolmogorov-Smirnov test and the results validated the null hypothesis with a mean error less than 12mm.

After solving the translational position of the mobile robot on a 2-dimentional plane, the mobile robot needs to be aware of its 3-dimentional orientation. To achieve this, a 9-axis sensor using an accelerometer, a gyroscope, and a magnetometer were implemented, to form an inertial measurement unit capable of returning a highly accurate self-orientation position using a directional cosine matrix which returns model free from accumulating error. A novel closed-loop PI controller was derived using the directional cosine matrix. In order to validate the directional cosine matrix method, data was collected from the sensor and compared against visual tracking data. The directional cosine matrix method data was validated using a non-parametric single-sample statistical analysis using the Kolmogorov-Smirnov test validated the null hypothesis with a mean error less than 1˚.

Now that the mobile robot has its position on a 2-dimensional plane, using the double compass sensor configuration method, and it has its 3-dimensional orientation, the mobile robot needs to move in an environment, without colliding with obstacles. This is achieved using computer vision, exploiting the RGB-D sensor from Microsoft, called the Kinect Sensor, were the RGB-D sensor provides an image of the environment and also depth data for the environment. This is important to enable the mobile robots to achieve critical tasks like environment map building and path planning. A machine learning method based on regularised linear regression was adopted to transform RGB-D sensor input into depth maps. Using this transformation, a novel method for transforming depth maps into two-dimensional environment map was presented in this thesis. In order to validation the method for converting the RGB-D data into distance, data from the sensor was collected and compared to empirically measured data. The compared data was validated using a non-parametric single-sample statistical analysis using the Kolmogorov-Smirnov test validated the null hypothesis with a mean error less than 2mm.

Computer vision was further implemented to solve the limitation of large, heavy, power hungry and computationally expensive camera systems on small mobile robots. A simplified stereo vision approach is highlighted which endeavours to solve these limitations by presenting a solution to transforming from an image point to an object point in three-dimensional space, using a projection matrix with intrinsic and extrinsic solutions and camera calibration methods.

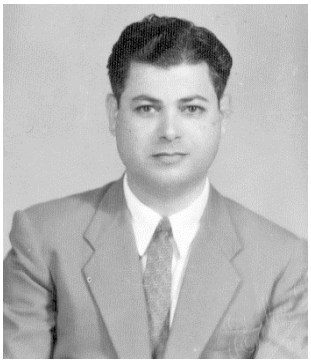
Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to acknowledge my late grandfather, Demosthenos Flatoridies, for inspiring me to pursue my PhD in Robotics.



I would like to thank my family, particularly my father, George Georgiou, and my loving mother, Helen Kasandra Georgiou, for their support during this long journey.

I would like to thank my employer at King's College London, Caroline Murphy, for her support and wisdom.

Finally, without the support of my supervisors, Professor Jian Dai and Professor Michael Luck, this research would not have been possible.

Chapter 1 Introduction

1.1 Background & Rationale

Mobile robots are getting more and more embedded into our daily lives, from robots that clean our floors every day, to those on far away planets exploring unknown environments, to those that save lives in hospitals transporting critical treatments to patients, and even those that we do not see going to places where humans cannot, to inspect pipes or in radiation hazardous environments that can harm humans. Mobile robots are important and they can contribute greatly to both our wellbeing, by saving lives, or even economically, by carrying out tasks faster than if they were done by a human manually.

Mobile robots are the most challenging research section in robotics engineering, starting with expert knowledge in mechanical systems, to design mobile robot system that have ability to move and can have controlled behaviours in unknown environments. It requires a deep understanding of electronics, to develop circuits and sensors that can interact and control these mechanical systems to control their behaviour. Furthermore, adept knowledge in software/firmware programming to write the code that will make these electrical systems control the mechanical systems, to achieve the tasks that are expected of these robotics systems.

The paper by Leonard et al [1], in 1990, summarizes the problem by getting the mobile robots to ask three critical questions: “Where am I?”, “Where am I going?”, and “How should I get there?”. The paper they wrote went into great detail about state-of-the-art sensors and technologies that aimed to answer these questions, but it is 2014 and we still do not have a perfect answer to any of those questions. The research focus for this thesis will mostly focus on the first question, as it is essential that a mobile robot knows where it is before it can know where it is going or how it is going to get there.

The progress in the field of nonholonomic wheeled mobile robots has not been as fast paced as expected from the interest generated and the advances made in the early days of research done in the mobile robot domain. In Voth’s paper, “Segway to the Future” [2], she starts her article with, “Imagine trying to solve a problem and being given just half the answer.” There are still a lot of tasks that are not possible because of limits in technology. The thesis presented here tries to answer some of those questions and bypass those limits in technology. The article references challenges in perception and obstacle avoidance, and the answers to this are

accurate self-localization and environment map building. Successful projects in the mobile robot domain have either operated in highly constrained and controlled environments or have been manipulated by including a level of human assistance. The results in the thesis are an un-manipulated analysis of results that shows the accuracy of self-localization for mobile robot navigation.

1.2 Problem Statement & Objectives

1.2.1 The Challenges

Navigation in unknown environments is a difficult task for mobile robots and has been taken for granted by humans and animals; they have no problems traversing in unfamiliar and new environments and can easily get from point A to point B, while also avoiding any obstacles that might or might not cross their path.

In mobile robot navigation research, there is strong likelihood to find the representation of localization features being described in Cartesian form (x, y, z). From the research carried out, it is still unclear if this is in fact the ideal or best way, or even the way humans or animals perceive such a low-level understanding of orientation of the localization features. The idea is to input information which is relevant to a specific task, while adding supervision when necessary via manual inputs, and let the mobile robot carry out the requested navigation based task. There has to be a better way, which has lead to exploiting genetic algorithms and artificial intelligence to try and overcome the failures of this approach. These new age methods are very appealing and go close to offering results in navigation systems, similar to those observed in nature.

Current research has returned a variety of successful solutions to perform restricted navigation based tasks, which range from distinct outcomes that deliver self-localization data to environment map building to map obstacles and generate trajectories. The difficulty and challenge is to combine environment maps with self-localization data. The outputs from these two systems are represented in two different distinct forms. For example, an environment mapping algorithm generates a gridded representation of space or the location of obstacles. The challenge is mapping the localization coordinates to the environment map data to be used by a closed loop controller for navigation. Brooks from MIT [3] writes an interesting paper, discussing the application of different behaviour modes of operation and sensing, which run concurrently in a form of layered architecture, and uses a voting mechanism to select an action towards a goal, rather than attempting to communicate these un-mapped representations between layers.

Finally, the last major challenge to bring up is the challenge of constructing the mobile robot body to be manoeuvrable, robust, and reliable to move around in these unknown, undetermined

environments. Powering robots imposes an added level of restriction to the design and leads to considering complex power management solutions, and it also adds limits to the time the mobile robot functions and performs tasks. Also, reprogramming and communicating with a computer that is not mounted on the robot with all of its sensors and control boards, are complex and convoluted processes.

In summary, mobile robot navigation is challenging because of the complexity of the world, even a controlled small indoor environment, has proven to be a challenge not yet mastered by robots because they do not have the capabilities of coping with every possible scenario they will encounter in unknown environments.

1.2.2 Self-Localization – Where am I?

All mobile robot devices are designed with the explicit purpose of navigating environments; this essentially is the most important role of the mobile robot. It is critical that a mobile robot avoids hazardous situations, including uncontrolled collisions with foreign objects or structures, and unsafe or unsuitable conditions, including exposure to unsuitable temperatures and radiation or even exposure to weather. Any mobile robot that wants to achieve the fundamental task of getting from point A to point B, has to be aware of where it is in its coordinate space and answer one of most common robotics questions, “Where am I?”, which is better technically defined as self-localization. The answer to the question “Where am I?” is typically resolved using one of two types of methods, a relative localization (dead-reckoning method) or an absolute localization.

Dead-reckoning methods are simple to implement but because it is solved as an iterative method and as a result of sensor accuracy, this method is prone to an accumulative drift of errors. This accumulative drift is mainly due to the system being setup with odometer sensors as incremental encoders on mobile robot wheels. The work presented by Yue et al [4] provides a good example of a relative self-localization model based on the dead-reckoning system. Being based on a two mobile robot which is in the manoeuvrable classification, this model is very relevant and shows how accumulative drift is a problem. It also attempts to solve the problem using ultrasonic sensor data.

The alternative to this accumulating error method is an absolute self-localization method, which is based on exteroceptive sensor data collection. The work by Arsenio and Ribeiro [5] is a perfect example of an absolute localization procedure for mobile robots using laser range finders for detecting natural landmarks for localization. The algorithms described in this paper for the absolute self-localization method are probabilistic and are based on the commonly used Kalman filter, which returns a highly accurate location estimate with a useful level of uncertainty of the computation.

The absolute self-localization method is highly accurate and does not suffer from accumulative error drifting, though this method is computationally expensive. While the relative self-localization method is not as accurate and suffers from a drifting error, it is less computationally expensive. So is the best approach relative or absolute self-localization? The answer is simple, use both methods and trade off their weaknesses. The special issue journal on mobile robots prepared by Borenstein et al [6] goes into extensive detail about the different types of self-localization methods and how well they work in the field. The idea presented is that the mobile robot needs the self-localization method to work on-line and to achieve this, relative self-localization is employed as the primary method with the absolute self-localization method run periodically to remove the drifting error from the computation.

1.2.3 The Objectives

The objectives of this thesis is to investigate self-localization ……. and verify the theory with a self-developed two-wheel manoeuvrable mobile robot platform that is capable collecting data from mechanical sensors, like shaft encoders, and inertial sensors, like an accelerometer, a magnetometer, and a gyroscope. The platform should be capable of storing and processing the data from the sensors, for analysis and validation.

Present a kinematic and dynamic model using the Lagrange D'Alembert principles on a two wheel manoeuvrable non-holonomic mobile robot. Derive the holonomic and non-holonomic constraints of the two wheel manoeuvrable non-holonomic mobile robot. To present a PD controller to control the motor inputs of the two wheels, of the manoeuvrable non-holonomic mobile robot.

Derive a method for estimating the mobile robots 2-dimensional position using the platforms wheel shaft encoders. Derive an analytical method for estimating the mobile robots 2-dimensional position using the wheel shaft encoders and a dual magnetometer configuration. Validate the dual magnetometer configuration using computer vision to track markers on the mobile robot. Present the descriptive and inferred statistical results, with a null hypothesis, to validate the dual magnetometer configuration against the computer vision data.

Derive an alternative method to the Kalman or complementary filter approaches, by a directional cosine matrix method for estimating the mobile robots 3-dimensional orientation using an inertial measurement unit composed of an accelerometer, magnetometer, and a gyroscope. Present a closed loop PD controller to manage the inputs from the accelerometer, the magnetometer, and the gyroscope and remove the accumulative drift of errors. Validate the directional cosine matrix method using computer vision to track the orientation of the sensor in a 3-dimensional space. Present the descriptive and inferred statistical results, with a null hypothesis, to validate the directional cosine matrix method against the computer vision data.

Derive a method for environment map building using the Microsoft Kinect RGB-D sensor. Present an analytical method for measuring distance from the data of the RGB-D sensor using a machine learning regularized linear regression approach. Validate the distance estimation method using empirically data. Present the descriptive and inferred statistical results, with a null hypothesis, to validate the analytical solution of machine learning approach for distance estimation.

1.2.4 Thesis Outline

The work in this thesis describes the progress made towards developing a mobile robot capable of tele-operated navigation, self-localization using mechanical and electrical sensors, and environment map building using vision based sensors. The work here is a blue print to building small mobile robots capable of complex navigational tasks providing detailed sensor feedback over the air.

In the following section, 1.2, the problem statement about the challenges of self-localization is looked at in some detail. Section 1.3 then covers the different known approaches for self-localization in small mobile robots.

The thesis is composed as follows:

Chapter 2 covers the background research on mobile robots and their different classifications. The manoeuvrable classification is covered in depth, with a look at mechanical and electrical sensors for self-localization. The chapter concludes with a look at visual sensors for environment map building.

Chapter 3 is a description of the hardware and software required to build the KCLBOT. The construction of the robot is covered in detail, from the electrical to mechanical, and finally the software for the firmware and the tele-operated device.

Chapter 4 introduces the nonholonomic mathematical models, both the kinematics and dynamics of the mobile robot and how the robot behaviour is optimized using Lagrange D'Alembert principles.

Chapter 5 describes a novel self-localization method using a hybrid configuration of mechanical sensors and digital electro-magnetic sensors. The results of how this configuration is effective are presented and compared against a single compass approach.

Chapter 6 presents an inertial measurement unit capable of sensing rotation and translation in 3 planes covering 9 degrees of freedom. The research into using a directional cosine matrix to carry out highly accurate self-localization is analysed and experimental results are presented.

Chapter 7 looks at the vision based RGB-D sensor and how it is exploited for environment map building. A model is presented using machine learning to optimize feature data to build a representation of distance which is used to build environment maps.

Chapter 8 describes the work carried out to build 3D world models based on a fixed plane to a projections plane, which is the early work required to enable stereo vision for mobile robots.

Chapter 9 concludes the thesis with a general discussion and ideas for future work.

1.3 Contributing Publications

The publications that contributed to this thesis are divided into three major categories that range from the two-wheel manoeuvrable mobile robot system model, to self-localization methods and computer vision concepts for small mobile robots.

1.3.1 Two-Wheel Manoeuvrable Mobile Robot System Modelling

[7] Experimental Study on Track-Terrain Interaction Dynamics in an Integrated Environment: Test Rig, “Proceedings of the Eleventh International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)”

[8] Lagrangian D’Alembert Modelled Manoeuvrable Autonomous Nonholonomic Mobile Robot Using a Closed Loop PD Controller, “ASME/IEEE 2009 International Conference on Mechatronic and Embedded Systems and Applications (MESA)”

[9] Regularized Linear Regression for Distance Estimation with an RGB-D Sensor, “International Conference on Autonomous Robot Systems and Competitions (Robotica)”

1.3.2 Self-localization Methods for Small Mobile Robots

[10] Self-localization of an autonomous manoeuvrable nonholonomic mobile robot using a hybrid double-compass configuration, “International Symposium on Mechatronics and its Applications (ISMA)”

[11] Using a dual compass configuration with shaft encoders for self-localization of an autonomous manoeuvrable nonholonomic mobile robot, “IEEE Conference on Robotics Automation and Mechatronics (RAM)”

[12] The KCLBOT: A Double Compass Self-localising Manoeuvrable Mobile Robot, “International Conference on Mechatronic and Embedded Systems and Applications (MESA)”

[13] The KCLBOT: A Framework of the Nonholonomic Mobile Robot Platform Using Double Compass Self-Localisation, “Mobile Robots – Current Trends”

[14] Self-localization using a 9DOF IMU Sensor with a Directional Cosine Matrix, “International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE)”

[15] Visual self-localization for nonholonomic mobile robots using a Hybrid Skip-list inspired Search Algorithm with a Gradient Policy, “IEEE Conference on Robotics Automation and Mechatronics (RAM)”

1.3.3 Computer Vision Concepts for Small Mobile Robots

[16] Mobile Robot Environment Map Building Using a RGB-D Sensor, “International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)”

[17] The KCLBOT: Exploiting RGB-D Sensor Inputs for Navigation Environment Map Building and Mobile Robot Localization, “International Journal of Advanced Robotic Systems (ARS)”

[18] A Stereo Vision Model for a Small Form Factor Autonomous Mobile Robot KCLBOT, “International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)”

[19] The KCLBOT: The Challenges of Stereo Vision for a Small Autonomous Mobile Robot, “International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE)”

Chapter 2 Background Research

2.1 Introduction

This chapter explores the background of mobile robot research and covers the following core research topics:

- The common applications of mobile robots
- Dead-Reckoning for mobile robots
- Inertial sensors for mobile robot navigation
- Computer vision methods for environment map building

Mobile robot research is a very large domain of research and the focus for this thesis is on the self-localization methods for small mobile robots and environment map building methods.

2.2 Applications for Mobile Robots

The mobility of mobile robots brings in the next frontier in flexible robotics, in which robots can change and adapt to unknown environments. That is not to say that fixed or static application mobile robots do not have a place because they will always have utility in manufacturing and other industries where non-dynamic systems suffice. The mobility of mobile robots offers additional flexibility to end users and also the promise of new applications. The applications range from, but are not limited to, medical or surgical applications, personal assistance, security and defence, manufacturing and distribution applications, and to go where humans cannot easily in ocean and space exploration. Some of these applications are explored below.

2.2.1 Medical and Surgical Mobile Robot Applications

Mobile robots can contribute to the wellbeing of humans and can assist with health critical operations to achieve this. Mobile robots have been given the service role of delivering various specimens from patient to laboratories and delivering medications to patients. The ER1 mobile robot [20], developed by Mohammad Saud Khan from ETH Zurich, is a perfect example of a mobile robot in a health care environment which is capable of performing tasking like path planning and obstacle avoidance. Mobile robots can also be used in surgical applications, using their mobility to diagnose or evaluate surfaces with sensors to find abnormalities. The work by Kane et al [21] presents a kinematic path, following the control model for mobile robots to achieve this. The mobile robot is used to traverse a skull and while doing so, builds a 3D surface map of the skull, which is then evaluated for abnormalities by a surgeon. These are just two examples of the utilisation of mobile robots in health care environments, but there is scope for further applications and possibilities.

2.2.2 Automated Inspection Mobile Robot Applications

Manual inspection is not a very resource-effective process, economically and otherwise. The manual inspection process is a very tedious procedure carried out by a human operator, and it suffers from the increased risk that human errors are not observed and are overlooked. Using mobile robots programmed with artificial intelligence for automated inspection will improve search times and return a better and controlled tolerance for finding abnormalities. The mobile

robot platform, developed by Wilson et al [22], is a good example of how an automated inspection mobile robot platform is optimized to search for and find abnormalities. Wilson et al developed a mobile robot platform with a computer vision camera system mounted onto the mobile robot's frame. The mobile robot is configured to automatically traverse a warehouse of drums containing low-level radioactive waste and toxic substances. The mobile robot uses computer vision to inspect changes in the drums to identify abnormalities, and raise a warning signal should any be found. This is a great example where mobile robots can be used for automated inspection because the radioactive waste is hazardous when prolonged human exposure is inflicted and the inspection by human observation will be more difficult than the computer vision system. Another good example of an automated inspection mobile robot is the robot presented by Choi et al [23]. This mobile robot is designed to navigate and inspect the out bends of pipes. The mobile robot is equipped with computer vision and comes with a forward and rear facing camera system, which are used for the inspection process and navigation. This is another clear example of why mobile robots are essential. Because of the narrow and difficult-to-reach nature of pipes, small mobile robots are ideally suited for this application.

2.2.3 Mobile Robots for Space Exploration

Mobile robots play a key role in space explorations because it is more economical to send a robot into space, rather than a human, because robots do not need oxygen, they take up less space, they carry tools humans cannot and they can perform exploration tasks without needing a break. The paper by Espinoza et al [24] is a good example of research in this domain, writing about novel and new methods for nonholonomic mobile robots tasked with exploring unknown environments. The research by Espinoza et al presents methods for building data structures using sensor-based random trees (SRT) that are used as road maps for mobile robot navigation in unknown environments.

These are just a few applications that mobile robots are capable of. The possibilities are limitless and as microcontrollers get smaller and more computational, powerful and more power efficient, more possibilities will open up. It is also worth noting the importance of sensors and how they open up possibilities in mobile robot research.

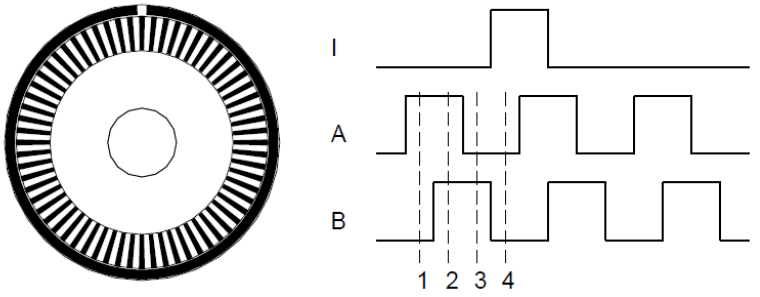
2.3 Mobile Robot Dead-Reckoning

The modern expression dead-reckoning is derived from the term “deduced reckoning”, which was used by sailors many years ago when electrical sensors did not exist. It is a simple mathematical concept that is used to determine the current location of a vessel by advancing from the previous position via a known course and velocity over a given duration of time [25]. It is very common to find that a majority of land-based mobile robot configurations are dependent on a dead-reckoning system as a component of the robot’s navigation strategy.

The simplest and most common implementation of the dead-reckoning system in robotics at this time is called odometry. The odometry is derived from the word ‘odometer’, which implies displacement along a path of travel commonly in vehicles. Normally in mobile robots, the term odometry refers to the instrumentation used, like optical encoders coupled onto the robot’s motor armatures or rotating wheel axles.

Optical encoders are inexpensive and integrate easily on mobile robots. The first optical encoders were developed in the mid-1940s by the Baldwin Piano Company, and the main purpose was for tone wheels that enabled electric organs to copy other musical instruments. Developing miniaturized optical encoders is both challenging and expensive, the paper written by Dimmler et al [26] provides a stratified list of encoder system to optimize the selection of an encoder system for small motors which are utilized in mobile robots.

The simplest form of the incremental encoder is normally a single-channel tachometer encoder. The idea is that a mechanical device is used to chop light that produces a number of sine or square wave pulses for each shaft revolution. The more pulses in the mechanical disk, the higher the resolution of the device. These types of mechanical sensor designs are relatively inexpensive and are ideal for velocity feedback applications in control systems. The only issue is when the sensor is used for extremely slow velocities. The paper by Nickson [27] shows that this is fundamentally due to quantization errors.



State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

Figure 2-1 An Incremental Optical Shaft Encoder

Figure 2-1 presents a typical incremental optical shaft encoder wheel with two channel square wave output [28]. The optical sensors for channels A and B are out of phase and are used to determine the direction of rotation with a phase-quadrature encoder. This configuration returns four unique output states $S_1 - S_4$ which are used to deduce the direction of rotation and when a movement has started and ended. A single slot on the outside ring of the disk, labelled I, is used to measure the full rotation of the disk.

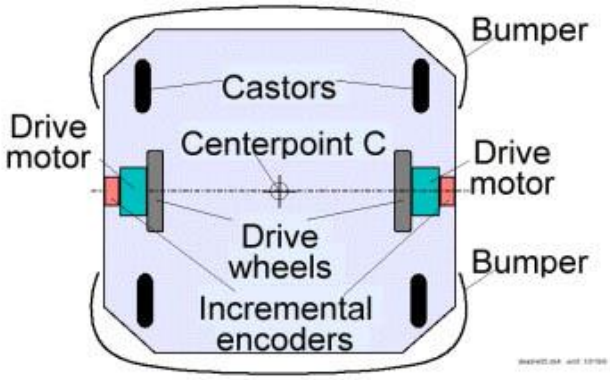


Figure 2-2 An Overview of a Differential-Drive Mobile Robot

The diagram above, Figure 2-2, presents a model of a typical differential-drive mobile robot. This mobile robot platform is called the *LabMate* and is manufactured by TRC. In this mobile robot configuration, the incremental encoders are mounted onto the two-wheel driving motors, which will be used to count the wheel revolutions. This mobile robot platform can perform dead-

reckoning by using simple geometric equations to compute the momentary position of the vehicle relative to the defined starting position.

The two best known resources for modelling this type of mobile robot system with odometry are the papers by Klarer [29], which provides a detailed model for 2-D navigation for wheeled mobile robots, and Crowley et al [30], whose paper is on controlling the rotation and translation of mobile robots. The following equations are well-known and give a simple model for implementing odometry for mobile robots.

Assume a sampling interval I where the left and right wheel incremental encoders return incremental pulse counts frequency increments N_L and N_R , respectively.

$$c_m = \pi.D_n / \mathrm{n}.\mathrm{C}_e \ (\mathrm{mm/pulse}) \tag{2.1}$$

Solving for c_m , where c_m is the conversion factor that translates the encoder pulses into linear wheel displacement. To achieve this, the nominal wheel diameter in mm (D_n), the encoder resolution in pulses per revolution (C_e), and the gear ratio of the reduction gear between the motor and the drive wheel (n), are required.

The incremental travel distance for the left and right wheel, $\Delta U_{L,i}$ and $\Delta U_{R,i}$ is computed as follows:

$$\Delta U_{L/R,i} = c_m.N_{L/R,i} \tag{2.2}$$

Which is followed by solving the incremental linear displacement of the robot's centre point C, denoted by ΔU_i, which is derived as follows:

$$\Delta U_i = (\Delta \mathrm{U}_R + \Delta \mathrm{U}_L) / 2 \tag{2.3}$$

Next, the robot's incremental change of orientation is computed as follows:

$$\Delta \theta_i = (\Delta U_R - \Delta U_L) / b \tag{2.4}$$

where b describes the wheelbase of the vehicle, which is ideally measured as the distance between the two contact points between the wheels and surface.

Next, the mobile robot's new relative orientation θ_i can be derived as follows:

$$\theta_i = \theta_{i-1} + \Delta\theta_i \tag{2.5}$$

Concluding with the mobile robot's centre point, this is derived as follows:

$$x_i = x_{i-1} + \Delta U_i \cos(\theta_i) \tag{2.6}$$

$$y_i = y_{i-1} + \Delta U_i \sin(\theta_i) \tag{2.7}$$

where x_i and y_i describe the mobile robot's relative position with reference to its centre point c at the instance i.

2.4 Mobile Robot Inertial Navigation

Having described dead-reckoning in the previous section, the inertial navigation approach is considered in this section. Inertial navigation was originally developed for aviation related applications but has been adapted to mobile robots. The inertial navigation technology has been adapted to use in aerospace, it can be found in missiles, and even has application in submarines. So, what exactly is inertial navigation? The principle idea is that it is a sensor platform that continuously senses acceleration data in three directional axes and integrates over time to derive velocity and position. This is typically achieved using a gyroscopically stabilized sensor platform to maintain consistent orientation of the three accelerometers during the changes in time.

The inertial navigation principle is fairly attractive because it is self-contained and does not require any external motion sensor data for positioning. The most important aspect of inertial navigation is the access to low latency dynamic orientation measurements. It is also important to note, as mentioned in the publication by Parish et al [31] on exterior autonomous navigation, that inertial navigation sensors normally have noise and errors that are independent from the external sensor. What this means is that the noise and error from the inertial navigation system is different from that of a dead reckoning system because it is self-contained, non-radiating, and most importantly non-jammable. An inertial navigation system is typically composed of a gyroscope, providing the angular rate, and an accelerometer, providing the velocity rate. While the sensor measurement can be highly accurate before the orientation can be deduced, the angular rate data must be integrated once and the linear velocity must be integrated twice to resolve orientation and linear position. As a result of the integration process, even very small errors in the sensor data can lead to an unresolvable accumulation in error.

2.4.1 The Accelerometer Sensor

The accelerometer is designed to measure proper acceleration. Proper acceleration is the physical acceleration observed by an object. The proper acceleration measured by an accelerometer is not necessarily the coordinate acceleration, which is described as the rate of change of velocity of an object. An accelerometer returns measurements in meters per second squared (m/s^2) or in G-forces (g). A single G-force on earth has the known equivalent of 9.8

m/s^2, but it should be noted that this can vary slightly with elevation. Accelerometers are generally used for sensing vibrations in systems or for orientation applications. Accelerometers are manufactured to have different sensitivities and tolerances to different gravitational ranges. Accelerometers are also designed to measure multiple axes. Normally, accelerometers are designed to work in 1g to 16g range, where the more sensitive 1g sensor would be used more for measuring tilt, whereas the less sensitive 16g sensor would be used to measure something like shock.

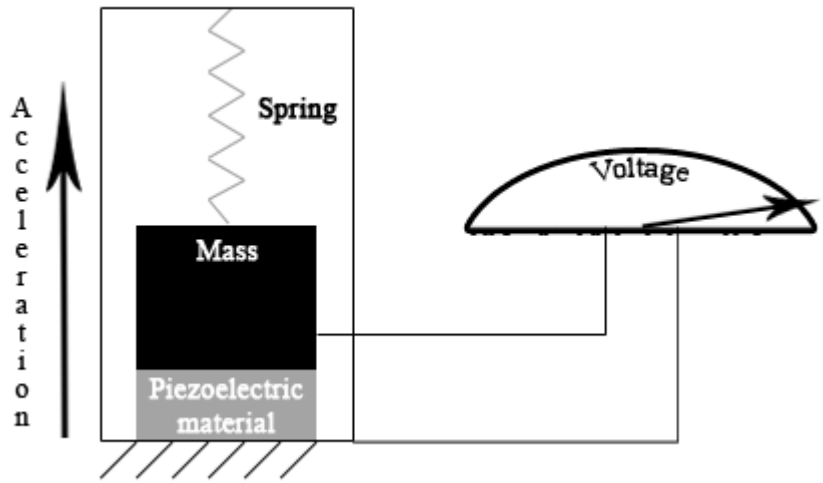


Figure 2-3 An example of the inside of a piezoelectric accelerometer

Typically, accelerometers contain capacitive plates internally. In some cases these are fixed, while in others they are attached to springs that move internally as acceleration forces act upon the sensor. As these plates move in relation to each other, the change in capacitance between them is what is used to determine the acceleration. Accelerometers can also be centred on piezoelectric materials, as depicted in Figure 2-3. The way it works is that the tiny crystal structures output electrical change when placed under mechanical stress.

For the research carried out for this thesis, micro electro-mechanical systems (MEMS), like the ADXL335 accelerometer sensor, were used. This particular accelerometer, which is manufactured by Analog Devices, is capable of measuring 3-dimensional accelerations, which covers acceleration along 3 axes. The sensor is sensitive to ±3g with a typical sensitivity of 300mV/g with a 3volt power supply.

Accelerometers are very easy to use and return fairly accurate results, but they are not perfect and can only perform and result measurement with low noise when the motion is carried out at high speeds and is free from vibrations. If the mass, depicted in Figure 2-3, is changing faster than sampling rate of the sensor, the results will be difficult to resolve accurately. The work published by Liu et al [32] provides a good reference on how to implement an accelerometer for mobile robot positioning.

2.4.2 The Gyroscope Sensor

A gyroscope is defined as a device that is used for measuring or maintaining orientation, which is based on the principles of angular momentum. The internal configuration of a gyroscope is designed as a spinning wheel or disc in which the axle is free to enable it to move in any orientation. This orientation does not remain fixed; it changes in response to an external torque. The device's orientation remains nearly fixed and is not coupled to the motion of the platform. The spinning wheel or disc can be measured in revolutions per second (RPS) or degrees per second ($^{\circ}/s$). Because this type of mechanical device is too difficult to miniaturize for this research, a MEMS gyroscope is implemented, which is based on electronic components.

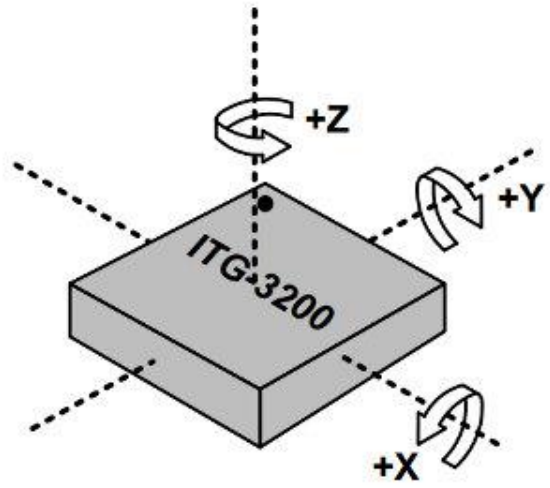


Figure 2-4 The ITG-3200 triple axis MEMS gyroscope

If you attach the ITG-3200 sensor, depicted in Figure 2-4, to a spinning wheel, the angular velocity aligned to sensor will be measure. A triple axis gyroscope, such as the ITG-3200, can measure rotation around three axes: x, y, and z. In the case of the spinning wheel, only the angle it's aligned with, will return an angular velocity, the other two axes would not measure any rotation.

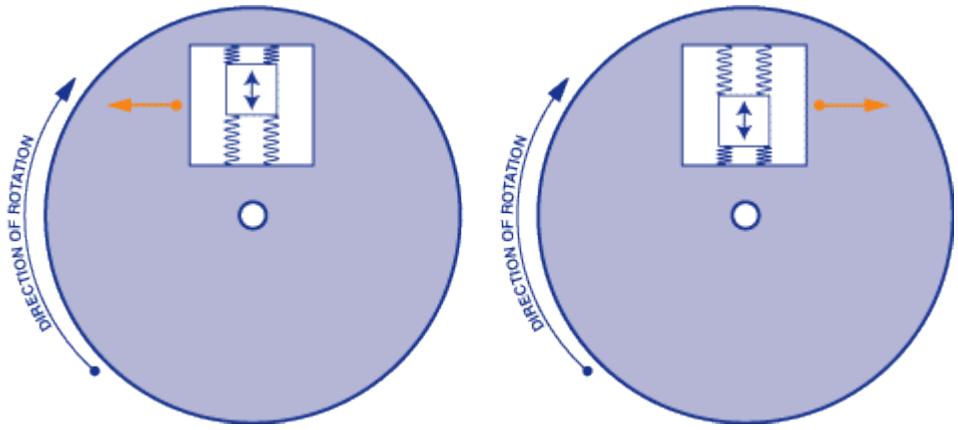


Figure 2-5 Internal operational view of a MEMS gyroscope sensor

A gyroscope sensor based MEMS can be very small and can be measured, in size, as small as 1 to 100 micro meters. The image in Figure 2-5, shows the internal operational view of the MEMS gyroscope sensor. When the gyroscope is rotating, a small resonating mass is shifted as the angular velocity changes. The rotating movement and the resultant shift in mass, is converted into very low-current electrical signals that is typically amplified and read by a microcontroller. The gyroscope value is then measured as the rate of change or the velocity of the angle. Just like the accelerometer measurements, the sensor has a sensitivity value provided by the manufacture and the values are returned as analogue voltage signals which need to convert by an ADC before they can be used. While the gyroscope offers a more accurate orientation value and is not subject to the same noise problem as the accelerometer. The paper by Myung et al [33] presents a model for implementing a gyroscope for mobile robot localization adapted on the Kalman filter.

2.5 Mobile Robot Computer Vision & Environment Map Building

In early November 2010, Microsoft released the Kinect™ sensor, which has the capability of RGB and RGB-D image capture. A camera system alone with RGB output alone does not provide enough information about the environment and only returns a 2D map of pixels with colours. When this optically matched image in integrated with depth information of the environment, this combination is referred to as RGB-D, where the D refers to depth. To provide the mobile robot with this extra information, the detail from the RGB-D camera is used to make a plot of the terrain, mapping the unobstructed space the mobile robot can utilize. Exploiting the pre-released drivers for this device, it is possible to export and manipulate both the RGB and RGB-D images for mapping, sensing, locating and modelling for SLAM.

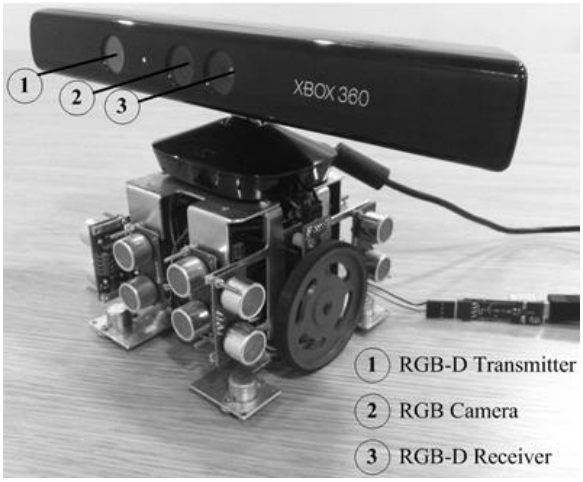
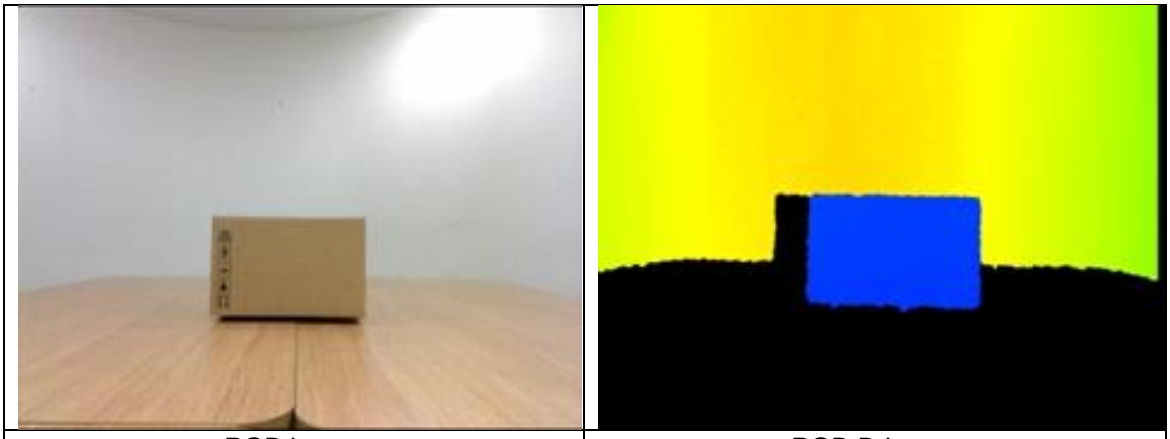


Figure 2-6 The Kinect™ Mounting Configuration on the KCLBOT

The Kinect™ depicted in Figure 2-6, which is mounted on the KCLBOT, is built around PrimeSense's PS1080 SoC [34]. The PS1080 acts as a controller for the IR light source in order to build the input image with IR light coding depth information. The light coding in this case is mapped on HSV [35]. The device is configured such that a standard CMOS image sensor receives the projected IR light source and transforms the image to produce a depth image. The sensor has a field of view of 58° Horizontal (H), 40° Vertical (V), and 70° Diagonal (D). The sensor allows for a maximum image depth size of VGA, which is a resolution of 640x480 and

has maximum image throughput frame rate of 60fps. Finally, the operational range of the sensor is limited to a minimum of 0.8m and to a maximum of 3.5m.



RGB Image	RGB-D Image

Figure 2-7 RGB and RGB-D Sensor Images

As depicted in Figure 2-7, the sensor is able to acquire RGB and RGB-D images, where the RGB-D images are used for environment map building.

Before the RGB-D images can be used for environment map building, the depth information from the image needs to be computed. To achieve this, a function needs to be derived to convert the RGB colour values to Hue. Once the Hue value is acquired, it can be used to compute relative distance estimation. Using the model presented by Joblove and Greenberg [35], the following equation is implemented. Hue is one of the main properties of colour, which is defined technically as the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, orange, yellow, green, blue, and violet.

Figure 2-8 Hue Scale

Where, Figure 2-8, presents a hue scale in the HSB/HSL encodings of RGB. Where, HSB is the hue, saturation, and brightness and HSL is the hue, lightness, and saturation.

$$H' = \begin{cases} undefined, & if\ C = 0 \\ \dfrac{G-B}{C} \bmod 6, & if\ M = R \\ \dfrac{B-R}{C} + 2, & if\ M = G \\ \dfrac{R-G}{C} + 4, & if\ M = B \end{cases} \tag{2.8}$$

In equation (2.8), the formula specifies the transpose of the hue value, H' , using $C = M - m$, which specifies the chroma value and where $M = \max(R, G, B)$ and $m = \min(R, G, B)$ specify the maximum and minimum values of the RGB colour, respectively.

The conversion function used to convert the transpose of Hue (2.8) to the Hue value needed, is as follows.

$$H = 60°.H' \tag{2.9}$$

Now that a conversion for Hue from RGB has been defined, a function to convert the Hue values to distance is required. To achieve this, sample images are captured from the sensor, using a static object with a known distance from the sensor.

Some of the major challenges in mobile robot research are path planning, obstacle avoidance and self-localization. For these challenges to be overcome, the mobile robot platform requires accurate and detailed data of the environment to navigate a path precisely, to avoid collisions with obstacles, and for an exact position and orientation of itself in a navigation space. The depth information from the RGB-D camera provides the data to enable the level of detail required for path planning, obstacle avoidance and self-localization.

Typically for small mobile robots, which have constraints on sensor processing, simple infrared or ultrasound range sensors are used to build environment maps and predicate obstructing obstacles. In Lee and Song's paper [36] on a rotating radar style IR range sensor for mobile robot localization, the result presented reveals the limitations of accurate environment map building using this type of range sensor configuration. As a consequence of the implementation, the mobile robot will also be limited by the scan time resolution. If the mobile robot is navigating

in an environment with dynamical changing obstacles, it will require a high refresh rate of the environment, to traverse a path accurately and to avoid obstacles.

An alternative to using an infra-red sensor is the use of a 3D laser range finder. In their paper, Surmann et al [37] implements an AIS 3D laser range finder to build navigation environment maps for the Ariadne robot platform. While the 3D laser range finder was capable of producing accurate data of the environment, this came at the cost of power consumption, which was rated at 17W with a 24V battery allowing 4.5Ah, and at the cost of the platform size to carry the battery and the scanner, which measured at 80cm x 60cm x 90cm.

An alternative approach to infrared or laser emitting systems is implementing a stereo camera system that can capture two images and produce a 3D environment with depth information. The PRISM [38] vision system, implemented on a mobile robot platform by Huber and Kortenkamp [39] as a real-time and high-bandwidth sensory system, is good example of an alternative approach. Huber and Kirtenkamp stress the importance of a vision system being capable of performing at a high bandwidth and delivering sensory data in real-time, or else the system will not be capable of performing important tasks in dynamic environments. The work by Haverinen et al [40] on a miniature mobile robot platform, using dual CMOS colour camera modules for a stereo vision system, offers an interesting approach to stereo vision platform but is limited as an autonomous platform because of local processing capabilities.

2.6 Summary & Conclusions

Mobile robots are dependent on sensors to function, just like humans are dependent on their sight to see and our sense of balance to complete the simple task of walking from one place to another. This thesis researches methods to exploit both dead-reckoning and inertial sensors to derive novel methods for self-localization. However, due to the use of numerical methods, to solve the localization problem, in dead-reckoning and due to the analogue signals from the sensors, such as the accelerometer, the magnetometer, and the gyroscope, this method is subject to accumulative drift, which leads to erroneous results in the mobile robot self-localizing.

This thesis presents an analytical solution that is less computationally expensive, without compromising the accuracy of self-localizing result of the mobile robot on a 2-dimentional plane. An alternative approach to the heuristic and memory intensive method of the Kalman filter or to the less accurate method of the complementary filter, which is presented using a directional cosine matrix method for estimating the mobile robots 3-dimentional orientation, using an inertial measurement unit, composed on an accelerometer, a magnetometer, and a gyroscope.

Mobile robots can benefit significantly from computer vision, giving them the ability to complete navigation task by having detailed data of their environment. This thesis presents research into using computer vision to build environment maps using the Microsoft Kinect RGB-D sensor and presents the analytical solution for distance estimation from the data of the sensor and how environment maps are constructed.

Chapter 3 The KCLBOT: A Manoeuvrable Nonholonomic Small Form Factor Mobile Robot

3.1 Introduction

The overall objective of this chapter is to deliver novel research into mobile robot platforms that specialize in self-localization and environment map building using mechanical and electrical sensors, developed by this candidate. The platform should be capable of being tele-operated and provide sensor data over the air wirelessly. The overall aim of the thesis was to carry out research in self-localization and environment map building rather than robot design, so this was taken into consideration to keep the system simple, while having multiple purposes, and will be easy to manage. The idea was to shift away from using highly specialised and expensive hardware in robotics sensors and computer vision and to move closer to using real world hardware, which can be used in homes and offices and can be built by end users with limited technical knowledge.

3.1.1 The System Requirements

The system requirements for the mobile robot can be categorized into the following areas:

- Manoeuvrable configured mobile robot platform in a small form factor
- Mechanical and electrical sensor acquisition and processing
- Power management and built-in re-chargeability
- Asynchronous wireless system operation and data acquisition
- Re-configurable microcontroller firmware
- Control software interface from a mobile device

Using the requirements defined above, a small form factor mobile robot, called the KCLBOT, was developed. The KCLBOT was first referenced in a conference paper in 2011 at the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference [12], which discussed the mobile robot novel approach to self-localization using a double compass configuration. In the same year, the KCLBOT was referenced in the International Journal of Advanced Robotic Systems (ARS) [17] as well as being discussed in the book chapter “Mobile Robots – Current Trends” [13], where the framework for the mobile robot was discussed in the journal paper, and environment map

building using a RGB-D sensor was featured in the book chapter. In 2012, the KCLBOT's stereo vision configuration was published in the conference proceedings of the Fifteenth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR) [18] and the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference [19].

The design for the KCLBOT is simplistic in appearance, but under the plastic exterior there are novel innovations and cutting edge technologies. The mobile robot's design is mostly based on commonly known Micromouse [34] competition robot designs.

To give a complete overview of the KCLBOT, Section 3.2 covers the overall design configuration of the mobile robot, Section 3.3 presents the electrical configurations and microcontroller systems, and Section 3.4 provides an outline of the firmware and software used to operate and control the mobile robot wirelessly.

3.2 Design Overview of Manoeuvrable Mobile Robot

The design for the KCLBOT is based on the manoeuvrable mobile robot configuration [41], which leads to a simple configuration of the rotating wheels parallel to each other.

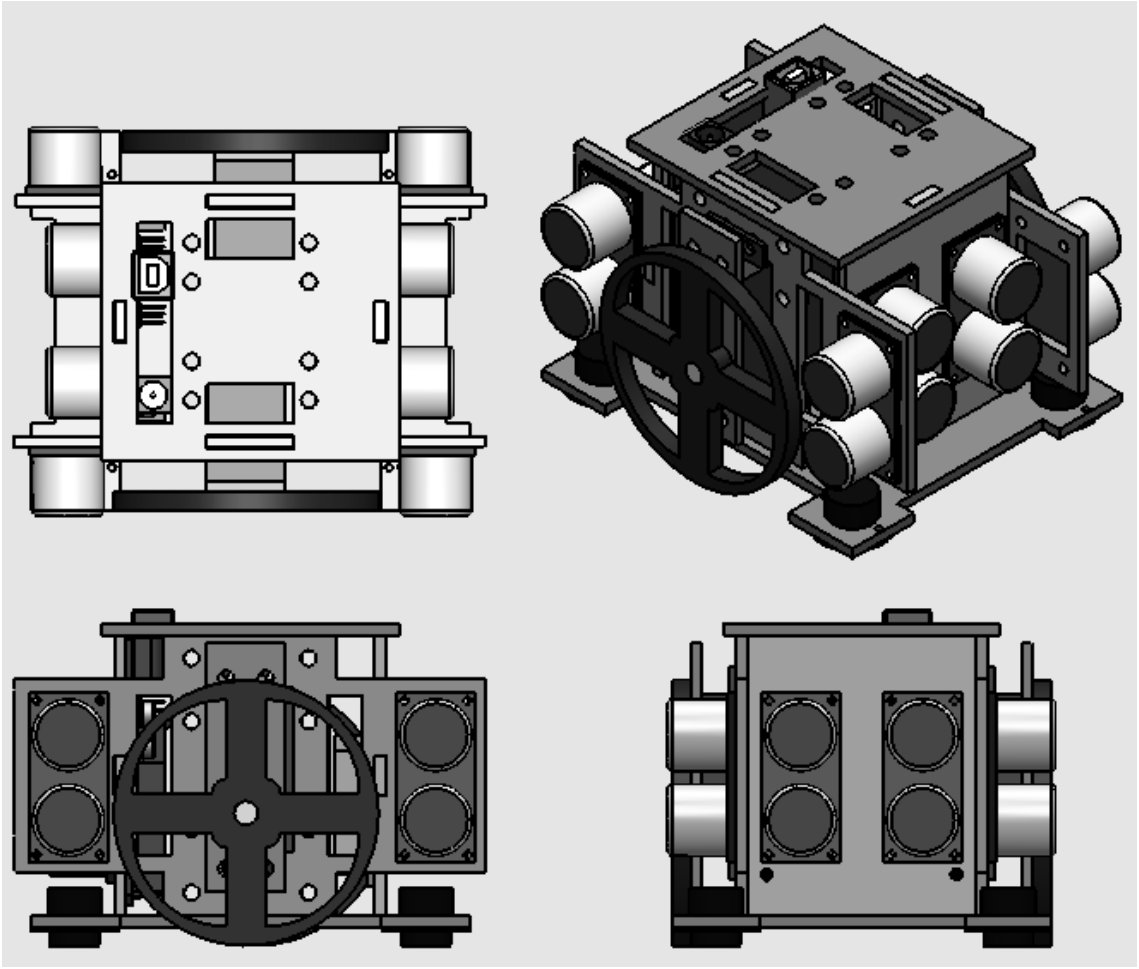


Figure 3-1 The KCLBOT Mobile Robot Design Overview

An overview of the KCLBOT is presented in Figure 3-1, showing the mobile robot from the top, the bottom, and the side. The housing for the mobile robot is made from acrylic plastic and is manufactured using a laser cutter [42]. The KCLBOT is configured with 8 ultrasonic range finder sensors [43] that are spaced proportionally in the four corners of the mobile robot's box design. The sensors are also placed at right angles to each other. This configuration enables an easier calculation when environment mapping algorithms are applied to the sensor data. The motors are actually continuous rotation servo motors with shaft encoders measuring the rotation of the wheels. Inside the mobile robot, mounts are provisioned for the wheel motion controller module and the main microcontroller module.

3.2.1 The KCLBOT Wheel Assembly

The drive system for the KCLBOT is made up of a wheel, a servo motor, and a quadrature encoder. The left and right side panels, as depicted in KCLBOT Designs 6, are used to hold the wheel and servo motor assemblies and the forward facing ultrasonic range sensors. The servo motor rotations are monitored by quadrature encoders [44] which are mounted directly onto the face of the servo motors and read data from inside of the rotating wheels. The quadrature encoders, WW-1 WheelWatcher Encoders, are manufactured by Nubotics and designed to form a dead-reckoning system [45]. The full design for side panels are presented in Appendix 12.1 - KCLBOT Designs 3.

3.2.2 The KCLBOT Main Control System

The main controller for the KCLBOT is what makes everything from motion to wireless communications happen. The rear facing panel, as depicted in KCLBOT Designs 7, is used to hold the Arduino compatible microcontroller [46], the Arduino sized shield with additional sensors, interfaces to the ultrasonic range sensors, Wi-Fi Communication system and LiPo batteries connected in series. The panel has the dual purpose of holding two ultrasonic range sensors as well. The full design for the rear facing panel is presented in Appendix 12.1 - KCLBOT Designs 5.

3.2.3 The KCLBOT Motion Control System

The motion controller interfaces with the main controller via a TWI interface and is responsible for precise movement and trajectory generation. The front facing panel, as depicted in KCLBOT Designs 8, is used to hold the motion controller, which is an interface to the servo motors, the quadrature encoders, and the power which comes from Arduino shield. The motion controller draws a regulated +5v for electronic circuit and +7.4v for the motors. The motion controller is manufactured by Nubotics, Wheel Commander WC-132, and with the encoders and servo motors, it makes a dead reckoning system. Like the rear facing panel, this panel also has a dual purpose to host ultrasonic range sensors. The full design for the front facing panel is presented in Appendix 12.1 - KCLBOT Designs 4.

3.2.4 The KCLBOT Design Summary

Using the principles and constraints for a manoeuvrable mobile robot with a simple box design, the KCLBOT is fashioned.

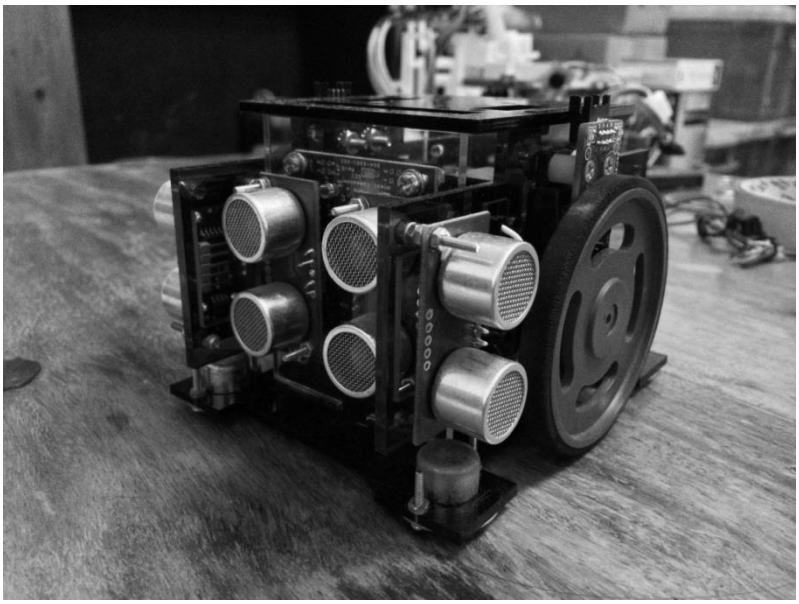


Figure 3-2 The KCLBOT - A Manoeuvrable Mobile Robot

Designing a mobile robot to conform to manoeuvrable constraints while also boasting wireless communication and cutting edge sensors is no simple task. The KCLBOT, as featured in Figure 3-2, does all those things. While the design of the mobile robot is simple, the hardware and features of the mobile robot are not.

3.3 Arduino Compatible Electrical and Electronic System

The mobile robot setup is sophisticated and bespoke/novel to the KCLBOT implementation. The mobile robot has the requirement to be powerful in a small package and offer the ability for wireless communication of sensor data feedback and remote tele-operated control.

3.3.1 The Arduino Compatible Atmega328p Microcontroller

To achieve this, the mobile robot requires a microcontroller capable of bandwidth data acquisition over digital/analog lines. The microcontroller architecture for the robot exploits the processing power from the ATMega328 microcontroller [47] with connection schematic presented in Appendix 12.2 - KCLBOT Electrical System 1. The microcontroller is based on the 8bit AVR architecture, and it is capable of running at a maximum clock frequency of 20MHz. The clock frequency of the KCLBOT implementation is set at 16MHz to support synchronization with the Arduino firmware [48]. The microcontroller comes with 32 Kbytes of flash, 2 Kbytes of SRAM, and 1023 Bytes of EEPROM memory which is ample for the Arduino boot-loader and the program to control the system. The microcontroller comes with 14 digital ports and 8 analog ports. For programming the device, 2 digital ports are used for receiving and transmitting data. The ultrasound sensors use 8 digital ports to communicate the sensor data. The dual shaft encoder configuration uses 2 analog lines to count the number of rotation ticks for each wheel and 2 digital lines to report the direction the wheel is moving in. For the daisy chained inertial measurement unit (IMU) and barometer sensor the two-wire interface (TWI) is used to communicate with these circuits using 2 analog lines.

3.3.2 Communication Configuration & Re-programmability

The microcontroller requires an interface to a computer, to program the firmware and implement the software that will control the KCLBOT. The ideal solution for this is the very commonly used USB data lines and micro USB interface. An interface to the ATMega328, as depicted in Appendix 12.2 - KCLBOT Electrical System 1, is required for programming and debugging software used to control the mobile robot. The FTDI FT232RL IC [49] acts as a bridge between the microcontroller and computer, communicating over a RS 232 serial interface. A special feature of the FT232RL is that it uses a common USB connector, which also allows the circuit to

draw power; the electronic configuration for the interface is shown in Appendix 12.2 - KCLBOT Electrical System 2. Having the USB power interface will enable the circuit to be powered, which is very useful for debugging and provides a mechanism to charge the battery for the mobile robot.

3.3.3 Microcontroller & System Power Management

The circuit has to handle and manage the load of microcontroller and the various IC's connected to the circuit, and having a stable power source that is not alternating or fluctuating is imperative to the stability of the circuit. For the mobile robot electrical system to function optimally, a clean and steady power supply is required. To achieve this, the electronic configuration employed in the figure found in Appendix 12.2 - KCLBOT Electrical System 3, is implemented. To smooth the power coming into the circuit, capacitors are placed between the input voltage and the ground. A voltage regulator is used to allow input voltages between 6v and 12v to power the electrical system. As an indicator that the system is receiving power, an LED is placed at the power output stage. To protect the system connected via USB to the circuit, a Schottky MBRA140 diode is used to prevent power flowing into the device connected.

The input voltage from the battery starts at 3.7 volts and can drop to 1v. The problem is that the circuit for the system requires a steady 5 volt stream of power. To resolve this issue, a voltage boost or step-up converter is required and the electronic configuration in Appendix 12.2 - KCLBOT Electrical System 4, is employed. This configuration exploits the NCP1402 IC by ON Semiconductors [50]. The configuration is employed to smooth the input voltage with a capacitor and block alternating current, only allowing direct current to pass. A diode is used across the input and output to prevent voltage from damaging the circuit and only allowing it to flow in one direction. The circuit also uses a capacitor on the output to smooth the output voltage. Since the mobile robot system is powered by a 3.7 volt at 1000mAh with a single cell Li-Polymer battery, an IC is required to charge the battery. The linear 500mA Microchip MCP73831 is used as a charge management controller [51], shown in Appendix 12.2 - KCLBOT Electrical System 5, to manage and charge the 1000mAh LiPo battery. An LED is used as a status indicator to show when the battery is fully charged and a capacitor is place on the battery to smooth the trickle of voltage going to the battery.

3.3.4 Integrated Sensor Electronic Circuit Configuration

For precision data acquisition results from the sensors, the circuit configuration needs to accommodate the extra draw on the power source and balance the power intake by adding smoothing capacitors at the power source and add pull-up resistors on the SDA and SCL data lines to the microcontroller.

A critical part of self-localization is using accurate sensor data and the ability to collect orientation sensor data at a high sample rate. Using a fast bus like the TWI will enable a sample rate as high as 100 kHz. The IMU configuration presented in Appendix 12.2 - KCLBOT Electrical System 6, measures orientation in 9 axes, which include 3 from accelerometer, 3 from the magnetometer, and 3 from the gyroscope sensor. The configuration is comprised of the ADXL345 accelerometer, the HMC5883L magnetometer, and the ITG-3200 MEMS gyroscope. The addition of a pressure sensor, shown in Appendix 12.2 - KCLBOT Electrical System 7, enables the mobile robot to have additional sensor data of the height orientation and the altitude of the system for localization [52]. This is done by exploiting the Bosch BMP085 high-precision, ultra-low power barometric pressure sensor. The pressure sensing range is between 300 to 1100hPa and in units of distance that is -500m to 9000m above sea level. The resolution of the sensor is up to 0.12 hPa/m. As the device is TWI capable, it easily daisy chains with the IMU sensor configuration.

3.3.5 Wireless Communication Circuit Configuration

Communicating with other external devices wirelessly is an essential requirement for the KCLBOT, and this is achieved by integrating wireless communications into the circuit configuration. Wireless communication is an essential component for the mobile robot [53], which is achieved using the Roving Networks RN-171 Wi-Fi module. The module communicates with the main microcontroller over UART serial; Appendix 12.2 - KCLBOT Electrical System 8 shows the schematic, the module incorporates an 802.11 b/g radio, TCP/IP stack, and a real-time clock.

3.3.6 Arduino Compatible Board with an add-on Shield

The circuit configuration is packaged into Arduino UNO R3 footprint to support the addition of a slave shield to extend the capabilities of the circuit and not have the limit for PCB space. The Arduino platform is widely used and is very popular for robotics implementations [54]. The circuit was designed around, as shown in Appendix 12.2 - KCLBOT Electrical System 9, the size and input/output constraints of the 3rd revision of the Arduino UNO.

The shield is attached to the main microcontroller board via same pin headers with the same input/outputs, as depicted in Appendix 12.2 - KCLBOT Electrical System 10, and this exploits the same form factor as the 3rd revision of the Arduino UNO. The shield is required because the PCB space is limited on main board and the system needs to host the IMU sensor, the BMP085 pressure sensor, the Roving Networks wireless module, an additional dual battery option, and 8 ultrasound range sensor headers with power and ground pins. The shield layer configuration leaves the configuration flexible to enable future additions to be layered onto the main microcontroller at a later stage.

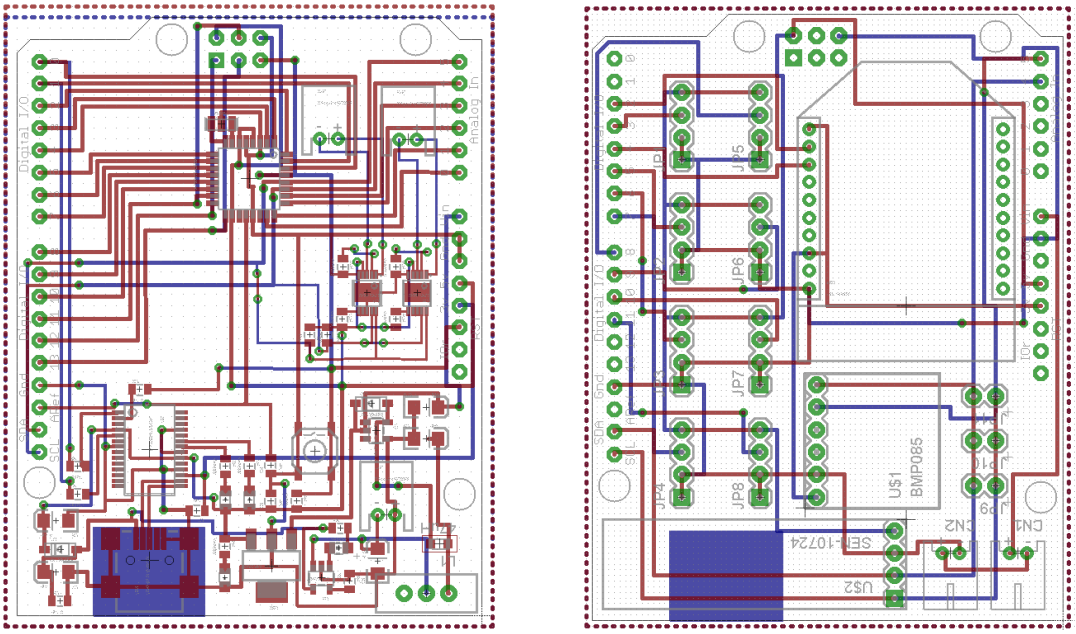
The ultrasound range sensors SRF05, manufactured by Robot Electronics, are interfaced using a single digital IO port and require 5 volts of power to operate. An example of a single sensor interface is depicted in Appendix 12.3 - KCLBOT Arduino Configuration 1. The sensors are ideal for small mobile robots because they can operate with only one digital port, rather than the typical configuration using two ports for trigger and echo. Another ideal feature for the sensor is that it has a minimum distance of 10mm which can be exploited for objects in a close proximity to the robot.

The 9-axis inertial measurement unit (IMU) sensor and barometer altitude sensor are interfaced to the microcontroller via the TWI bus using the dedicated SDA and SCL ports. The connection interface is depicted in Appendix 12.3 - KCLBOT Arduino Configuration 2. Both sensors are manufactured by Sparkfun and both require 5 volts to operate.

The wireless transmitter and receiver, manufactured by Roving Networks, is interfaced using the microcontroller serial transmitter and receiver lines. The interface to microcontroller is depicted in Appendix 12.3 - KCLBOT Arduino Configuration 3 with an additional toggle switch that allows the device to be mechanically configured to a wireless ad-hoc mode.

3.3.7 Microcontroller PCB Overview

The circuit configuration is based on the major sections described above- the main Arduino compatible microcontroller, communications & re-programmability, system power management, integrated sensors, and wireless communications.



The Microcontroller Board The Sensor Shield

Figure 3-3 Microcontroller & Shield PCB Design

Custom printed circuit boards (PCB) were designed to host the electronic configurations and circuit schematics; these are depicted in Figure 3-3. The design constraints for both PCBs were achieved using the Arduino UNO revision 3 specifications and dimensions. The boards were designed using 2 layers and routing from the top to bottom layer using vias. The board design exploited surface mount technology (SMT) over through hole technology. SMT IC's are smaller than through IC's and also generally use less power, which enables the design footprint to conform to the Arduino UNO R3 size and configuration.

3.3.8 Dual Motor Motion Controller System

Controlling two independent servo motors is a challenging task that requires critical power management to drive the servo motors smoothly with encoder feedback; this will require a microcontroller of its own to achieve this task.

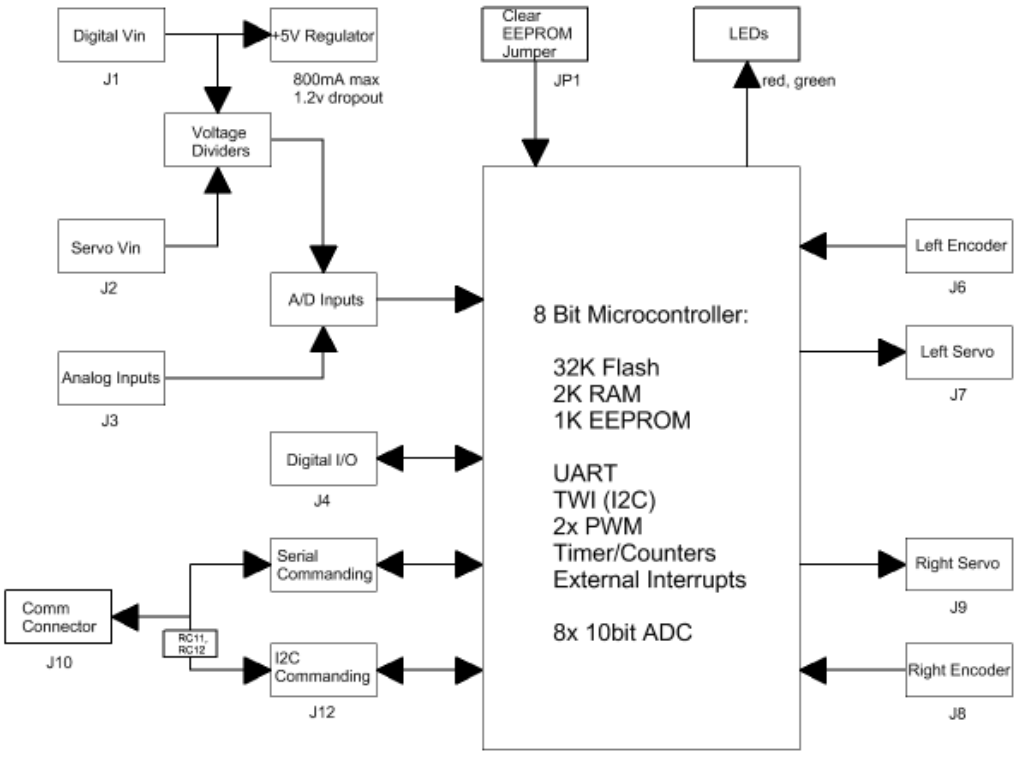


Figure 3-4 The Motion Controller System

The mobile robot requires an optimized system controller that will output a smooth rotation motion to both the mobile robot's wheels. This is achieved using the input from the mobile robot's left and right shaft encoders. To achieve this, a system control system is utilized, the Nubotics [55] WC-132 WheelCommander controller with 2 the WW-01 WheelWatcher encoders were selected to control the mobile robot's motion. This controller acts as a closed loop differential drive controller using 2 standard servo motors and the feedback from the 2 wheel encoders as shown in Figure 3-4. The motion controller communicates with main microcontroller using RS 232 serial or the TWI interface.

3.4 Summary & Conclusions

Mobile robots are very complex systems and the KCLBOT is a valuable contribution to mobile robot research. This chapter covers the following major challenges required to be addressed to develop a mobile robot platform capable of self-localization research and environment map building:

- Mechanical design
- Electrical systems
- Software integration

At the beginning of the chapter, clear system requirements were specified, including a manoeuvrable configured mobile robot platform in a small form factor, mechanical and electrical sensor acquisition and processing, power management and built-in re-chargeability, asynchronous wireless system operation and data acquisition, re-configurable microcontroller firmware, and control software interface from a mobile device. All of these requirements have been satisfactorily responded to in this chapter, resulting in the development of the KCLBOT mobile robot platform.

The novel mechanical design for the KCLBOT is both simple yet functional, with the capability to hold a very complex array of sensor for research in self-localization. The electrical system delivered a novel control board with cutting edge functionality, like wireless communication for bi-directional data communication, on-board power management, and a dead-reckoning drive mechanism. The firmware is equally important as it is core for integrating the mechanical system with the electrical system.

Chapter 4 Lagrange D'Alembert Modelled Mobile Robot with a PD controller

4.1 Introduction

In the previous chapter, Chapter 3, the design of the KCLBOT was explored in significant detail. Chapter 3 presented how the KCLBOT mobile robot system is constructed, both in terms of its mechanical and electronic configuration, and using a manoeuvrable nonholonomic mobile robot classification. There are a number of different types of mobile robot classification (i.e. manoeuvrable, car-like, omni-directional, etc.) and all of these classifications have their benefits and limitations. The car-like configuration is typically configured with three or four wheels with a steering mechanism used to direct the platform in the direction of navigation. The steering mechanism makes the car-like configuration complicated to design and control and leads to constraints in movement. In contrast, the omni-directional configuration is typically built with three or four wheels and has the ability to move in any direction without changing pose. The omni-directional configuration is relatively easy to design and is known to be complicated to control, but benefits from omni-directional movement. The manoeuvrable configuration is typically built with two moving wheels and occasionally comes with caster wheels to balance the mobile robot. The manoeuvrable configuration is easy to construct with only two wheels. While it has some constraints in movement, needing to rotate about its central axis to change pose, it is relatively uncomplicated to control. Reviewing the different possible configurations, the manoeuvrable configuration was the most balanced selection owing to a simple design with two wheels with manageable constraints.

In this chapter, Chapter 4, the constraints of the mobile robot are defined with the kinematic and dynamic model of the KCLBOT, along with a controller with optimization for the mobile robot platform. Research into understanding mobile robot structures is crucial and of active interest, and this research will contribute to the better understanding of the behaviour of manoeuvrable classified mobile robots. Alongside this, research into the optimization of the behaviour of mobile robots is also very important. This chapter provides a model and an optimized controller to help improve the behaviour of the manoeuvrable mobile robot classification.

The mobile robot is designed to move from point A to point B to achieve its primary goal of being mobile. Ideally, the mobile robot is required to get from point A to point B as quickly as possible. To achieve this behaviour of getting from one point in space to another as quickly as possible, a controller is required that will manage the outputs to the two wheels of the manoeuvrable configured mobile robot. To develop an optimized controller for the KCLBOT, the following steps need to be achieved:

- Define the systems holonomic and nonholonomic constraints and singularities.
- Derive the systems equation of motion using the Lagrangian of the known system.
- Use the Euler-Lagrange equation with Lagrange multipliers to optimize the system response.
- Using closed-loop control methodology derive a proportional-derivative controller.

The controller is essential for the mobile robot to perform the critical task of getting from point A to point B. The optimization of the controller is critical to the behaviour of the system completing the path following task as quickly as mechanically and physically possible.

The scope for this research is focussed only on the manoeuvrable two-wheel mobile robot with four supporting caster wheels, to balance the mobile robot, and the rotation motion being about the central axis of the mobile robot. The research here does not discuss tracked vehicles or configuration with different caster wheel setups, nor does it discuss manoeuvrable configurations where the centre of gravity is offset from the central axis point of the mobile robot. It is assumed that the mobile robot is nonholonomic in behaviour and for the purpose of modelling the mobile robot; it is assumed that the wheels of the system do not suffer from slip.

4.2 System Constraints of a Manoeuvrable Mobile Robot

In the manoeuvrable classification of mobile robots [56], the vehicle is defined as being constrained to move in the vehicle's fixed heading angle. For the vehicle to change manoeuvre configuration, it needs to rotate about itself.

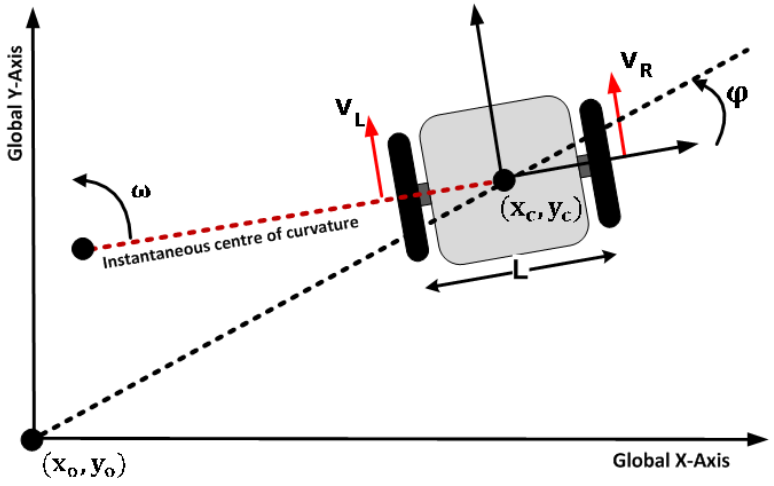


Figure 4-1 A typical two-wheel manoeuvrable mobile robot

As the vehicle traverses the two dimensional plane as presented in Figure 4-1, both left and right wheels follow a path that moves around the instantaneous centre of curvature at the same angle, which can be defined as the angular rate ω, and thus the angular velocity of the left and right wheel rotation can be deduced as follows:

$$V_L = \omega(icc_r - \frac{L}{2})\ mm/\text{s} \tag{4.1}$$

$$V_R = \omega(icc_r + \frac{L}{2})\ \text{mm/s} \tag{4.2}$$

where L is the distance between the centres of the two rotating wheels, and the parameter icc_r is the distance between the mid-point of the rotating wheels and the instantaneous centre of curvature. Using the velocities equations (4.1) and (4.2) of the rotating left and right wheels, V_L and V_R linear velocities respectively, the instantaneous centre of curvature, icc_r and the curvature angle, ω can be derived as follows:

$$icc_r = \frac{L(V_R + V_L)}{2(V_R - V_L)} \text{ mm} \tag{4.3}$$

$$\omega = \frac{(V_R - V_L)}{L} \ (\text{s}^{-1}) \tag{4.4}$$

Using equations (4.3) and (4.4), two singularities can be identified. When $V_R = V_L$, the radius of instantaneous centre of curvature, icc_r tends towards infinity, and this is the condition when the mobile robot is moving in a straight line. When $V_R = -V_L$, the mobile robot is rotating about its own centre and the radius of instantaneous centre of curvature, icc_r, is null. When the wheels on the mobile robot rotate, the quadrature shaft encoder returns a counter tick value; the rotation direction of the rotating wheel is given by positive or negative value returned by the encoder. Using the number of tick counts returned, the distance travelled by the rotating left and right wheels can be deduced in the following way:

$$d_L = \frac{L_{ticks}\pi D}{L_{res}} \text{ mm} \tag{4.5}$$

$$d_R = \frac{R_{ticks}\pi D}{R_{res}} \text{ mm} \tag{4.6}$$

where $\mathrm{L}_{\mathrm{ticks}}$ and $\mathrm{R}_{\mathrm{ticks}}$ depicts the number of encoder pulses counted by left and right wheel encoders, respectively, since the last sampling, and D is defined as the diameter of the wheels. With resolution of the left and right shaft encoders $\mathrm{L}_{\mathrm{res}}$ and $\mathrm{R}_{\mathrm{res}}$, respectively, it is possible to determine the distance travelled by the left and right rotating wheels, d_{L} and d_{R}. This calculation is shown in equations (4.5) and (4.6).

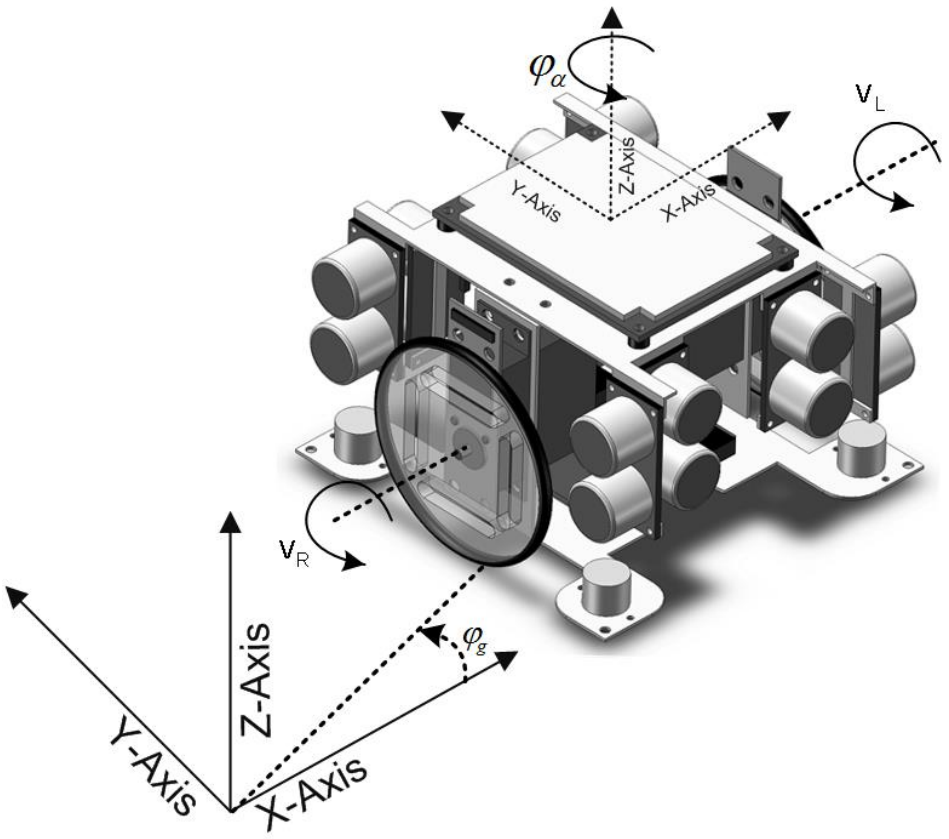


Figure 4-2 A manoeuvrable nonholonomic mobile robot

In the field of robotics, holonomicity [57] is demonstrated as the relationship between the controllable and total degrees of freedom of the mobile robot, as presented by the mobile robot configuration in Figure 4-2. In this case, if the controllable degrees of freedom are equal to the total degrees of freedom, then the mobile robot is defined as holonomic. Otherwise, if the controllable degrees of freedom are less than the total degrees of freedom, it is nonholonomic. The physical reason that the system is nonholonomic, is that the points on the common wheel axis cannot move laterally and as such this contributes to a degree of freedom loss. The manoeuvrable mobile robot has three degrees of freedom, which are its position in two axes and its orientation relative to a fixed heading angle. This individual holonomic constraint is based on the mobile robot's translation and rotation in the direction of the axis of symmetry and is represented as follows:

$$\dot{y}_c \cos(\phi_g) - \dot{x}_c \sin(\phi_g) - d\dot{\phi}_g = 0 \tag{4.7}$$

where, x_c and y_c are Cartesian-based coordinates of the mobile robot's centre of mass, which is defined as P_c, and ϕ_g describes the heading angle of the mobile robot, which is referenced from the global x-axis, and d is the deviation distance of the mobile robots centre of gravity. In classical mechanics a system is defined as holonomic if all constraints of the system are holonomic. For the constraints to be holonomic, the system must be expressible as a function. The holonomic constraints can only depend on coordinates and/or time and must not depend on velocities. To conclude, Equation (4.7) presents the pose of the mobile robot. The mobile robot has two controllable degrees of freedom, which control the rotational velocity of the left and right wheel and, adversely – with changes in rotation – the heading angle of the mobile robot is affected; these constraints are stated as follows:

$$\dot{y}_c \sin(\phi_g) + \dot{x}_c \cos(\phi_g) + L\dot{\phi}_g = rV_R \tag{4.8}$$

$$\dot{y}_c \sin(\phi_g) + \dot{x}_c \cos(\phi_g) - L\dot{\phi}_g = rV_L \tag{4.9}$$

where V_R and V_L are the linear displacements of the right and left mobile robot wheels, respectively, and where r describes the radius of the mobile robot's driving wheels. As such, the two-wheeled manoeuvrable mobile robot is a nonholonomic system. To conclude, Equation (4.8) and (4.9) describe the angular velocity of the mobile robot's left and right wheel.

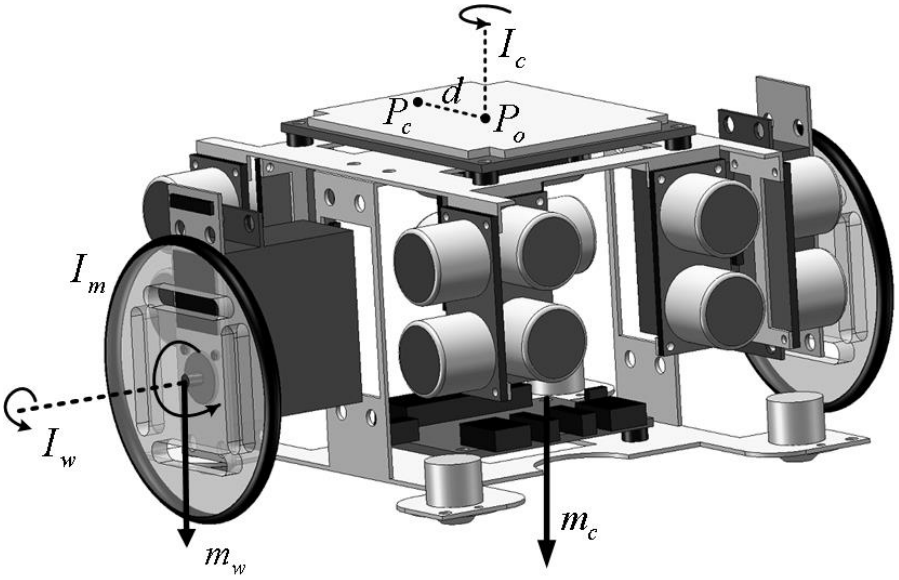


Figure 4-3 The Mobile Robot Drive Configuration

Symbol	Description of Structured Constant
P_o	The intersection of the axis of symmetry with the mobile robot's driving wheel axis
P_c	The centre of the mass of the mobile robot
d	The distance between P_o and P_c
I_c	The moment of inertia of the mobile robot without the driving wheels and the rotating servo motors about a vertical axis through P_o
I_w	The moment of inertia of each of the wheels and rotating servo motors about the wheel's axis
I_m	The moment of inertia of each of the wheels and rotating servo motors about the diameter of the wheels
m_c	The mass of the mobile robot without the driving wheels and the rotating servo motors
m_w	The mass of each of the mobile robot's wheels and rotating motors

Table 4-1 The Mobile Robots Constants

Based on the mobile robot drive configuration, presented in Figure 4-3, Table 4-1 describes the structured constants required to provide the physical characteristics of the mobile robot's movement.

4.3 Equation of Motion and Optimization with Lagrange Multipliers

Using the diagrammatic model expressed by Figure 4-2 and Figure 4-3, and the structured constants listed in Table 4-1, the nonholonomic equations of motion with Lagrange multipliers are derived using Lagrange – d'Alembert's principle [58] and are specified as follows:

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = E(q)u + B^T(q)\lambda_n \tag{4.10}$$

where $\mathrm{M(q)}$ describes an $\mathrm{n \times n}$ dimensional inertia matrix, and where $\mathrm{M(q)\ddot{q}}$ is represented as follows:

$$\begin{bmatrix} (\mathrm{m_c + 2m_w}) & 0 & -\mathrm{m_c d}\ddot{\phi}\sin(\phi) & 0 & 0 \\ 0 & (\mathrm{m_c + 2m_w}) & \mathrm{m_c d}\ddot{\phi}\cos(\phi) & 0 & 0 \\ -\mathrm{m_c d}\sin(\phi) & \mathrm{m_c d}\cos(\phi) & \mathrm{I_z} & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I_w} & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I_w} \end{bmatrix} \begin{bmatrix} \ddot{\mathrm{x}}_\mathrm{c} \\ \ddot{\mathrm{y}}_\mathrm{c} \\ \ddot{\phi}_\mathrm{c} \\ \dot{\mathrm{V}}_R \\ \dot{\mathrm{V}}_L \end{bmatrix} \tag{4.11}$$

Here, $\mathrm{I_z = (I_c + 2m_w(d^2 + L^2) + 2I_m)}$ and $\mathrm{V(q,\dot{q})}$ describes an n dimensional velocity dependent force vector and is represented as follows:

$$\begin{bmatrix} -m_c d\dot{\phi}^2 \cos(\phi) \\ -m_c d\dot{\phi}^2 \sin(\phi) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.12}$$

$\mathrm{G(q)}$ describes the gravitational force vector, which is null and is not taken into consideration, u describes a vector of r dimensions of actuator force/torque, $\mathrm{E(q)}$ describes an $\mathrm{n \times r}$ dimensional matrix used to map the actuator space into a generalized coordinate space, and $\mathrm{E(q)u}$ is specified as follows:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{4.13}$$

Where the expression is in terms of coordinates $(\mathrm{q},\dot{\mathrm{q}})$ where $\mathrm{q} \in \mathbb{R}^n$ is the position vector and $\dot{\mathrm{q}} \in \mathbb{R}^n$ is the velocity vector. Where q is defined as $[\mathrm{x_c},\mathrm{y_c},\phi,\mathrm{V}_R,\mathrm{V}_L]^{\mathrm{T}}$, the constraints equation can be defined as $\mathrm{A}(\mathrm{q})\dot{\mathrm{q}} = 0$. Where, $\mathrm{A}(\mathrm{q})$, the constraints matrix is expressed as follows:

$$A(q)\dot{q} = \begin{bmatrix} -\sin(\phi) & \cos(\phi) & -d & 0 & 0 \\ -\cos(\phi) & -\sin(\phi) & -b & r & 0 \\ -\cos(\phi) & -\sin(\phi) & b & 0 & r \end{bmatrix} \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\phi} \\ V_R \\ V_L \end{bmatrix} \tag{4.14}$$

Where finally, $\mathrm{B}^{\mathrm{T}}(\mathrm{q}) = \mathrm{A}^{\mathrm{T}}(\mathrm{q})$ and λ_{n} describes an m dimensional vector of Lagrange multipliers and can be described as follows:

$$\begin{bmatrix} -\sin(\phi) & -\cos(\phi) & -\cos(\phi) \\ \cos(\phi) & -\sin(\phi) & -\sin(\phi) \\ -d & -L & L \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{4.15}$$

One of the most common problems in calculus is that of finding the maxima or minima of a function, but it is often difficult to find a closed form for the function being extremised. This is an issue when it is required to maximize or minimize a function subject to fixed constraints. The Lagrange multipliers are used a tool to solve this problem without the need to explicitly solve the constraints and use them to eliminate extra variables. Equation (4.10) describes the Lagrange representation of the KCLBOT, in a state-space model, and Equations (4.11), (4.12), (4.13), (4.14), and (4.15) decompose the state-space model.

4.4 Closed Loop Feedback with a PD Controller

By using the equation of motion (4.10) and the constraints equations (4.7), (4.8), and (4.9), in state space, a state vector is chosen as $S(q)$, such that $A(q).S(q) = 0$. This can be seen as follows:

$$S(q) = \begin{bmatrix} \frac{r(b\cos(\phi) - d\sin(\phi)}{2b} & \frac{r(b\cos(\phi) + d\sin(\phi)}{2b} \\ \frac{r(b\cos(\phi) + d\sin(\phi)}{2b} & \frac{r(b\cos(\phi) - d\sin(\phi)}{2b} \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.16}$$

Since $\dot{q}$ from the constraint equation (4.14) is in the null space of $A(q)$ and because the columns of $S(q)$ are in the null space of $A(q)$ and are linearly independent, $\dot{q}$ can be expressed as the linear combination of the two columns of $S(q)$, such that:

$$\dot{q} = S(q)v \tag{4.17}$$

The reason for the introduction of v is to setup two independent velocity variables to drive the mobile robot, such that:

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} V_R \\ V_L \end{bmatrix} \tag{4.18}$$

By differentiating $\dot{q}$ in equation (4.17) and substituting the resultant $\ddot{q}$ into the equation of motion, equation (4.10) and by multiplying by the transpose of $S(q)$, the following is deduced:

$$S^T(M S\dot{v}(t) + M\dot{S}v(t) + V) = \tau \tag{4.19}$$

With the state space vector $x = \begin{bmatrix} q^T & v^T \end{bmatrix}^T$, where $x = \begin{bmatrix} x_c & y_c & \phi & V_r & V_l & \dot{V}_r & \dot{V}_l \end{bmatrix}$, which represents the motion and constraints equations of the mobile robot in state space.

$$\dot{x} = \begin{bmatrix} Sv \\ \dfrac{-(S^T MSv + S^T v)}{S^T MS} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{S^T MS} \end{bmatrix} \tau \tag{4.20}$$

This can be further simplified by applying the nonlinear feedback $\tau = S^T MS\left(u - \dfrac{-S^T MSv - S^T V}{S^T MS}\right)$.

$$\dot{x} = \begin{bmatrix} Sv \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{4.21}$$

Now that there is a clear model of the mobile robot based on the simplified version of the equation of motion and constraints equation, a control algorithm can be implemented on equation (4.21) and it can be assumed that u is the input to our system and as such, the next step is the deduction of u. The idea starts by differentiating our system output as follows:

$$y = h(q) = \begin{bmatrix} x_r & y_r \end{bmatrix}^T \tag{4.22}$$

Where x_r and y_r are the mobile robot's relative position. Now using equation (4.17) the following is deduced:

$$\dot{y} = \left(\frac{dh(q)}{dq}\right)\dot{q} = J_k(Sv) = (J_k S)v = \Phi v \tag{4.23}$$

Where Φ is defined as the decoupling matrix and can be represented as follows:

$$\Phi = J_k(q).S(q) = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \tag{4.24}$$

Where the 2x2 matrix is better described as follows:

$$\begin{bmatrix} \dfrac{r((b-y_r)\cos(\phi) - (d+x_r)\sin(\phi))}{2b} & \dfrac{r((b+y_r)\cos(\phi) + (d+x_r)\sin(\phi))}{2b} \\ \dfrac{r((b-y_r)\sin(\phi) + (d+x_r)\cos(\phi))}{2b} & \dfrac{r((b+y_r)\sin(\phi) - (d+x_r)\cos(\phi))}{2b} \end{bmatrix} \tag{4.25}$$

Now that the decoupling matrix is decomposed above, equation (4.24) is differentiated as follows:

$$\ddot{y} = \dot{\Phi} v + \Phi \dot{v} \tag{4.26}$$

$$u = \frac{v - \dot{\Phi} v}{\Phi} \tag{4.27}$$

Now that the system input u is defined in equation (4.27) the control equation is constructed as follows:

$$\ddot{y} = v = \ddot{y}_d + k_d(\dot{y}_d - \dot{y}) + k_p(y_d - y) \tag{4.28}$$

Where y_d is the desired path chosen for the mobile robot to follow and the feedback error is defined as $e = y_d - y$. Where y is the mobile robot's actual position and k_d and k_p represent the differential and proportional gain to our system, respectively.

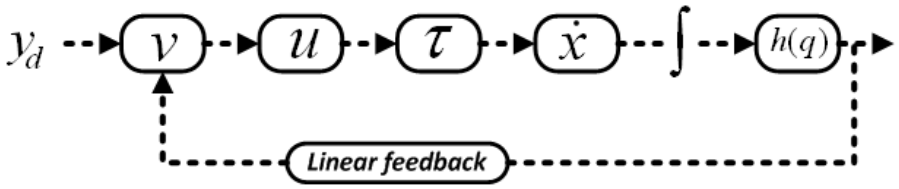


Figure 4-4 PD Controller with Linear Feedback

The diagram above, Figure 4-4, represents the control structure for motion of the mobile robot about a desired path. The control structure starts with the input of y_d, which is the desired path to be followed, and this is fed into v, equation (4.18), which is then used to calculate u, equation (4.27), and is deduced using the decoupling matrix Φ, equation (4.25), and now that u is defined, τ can be deduced, equation (4.20), and this is then used to calculate $\dot{x}$, equation (4.21), which is then integrated and represents $h(q)$, equation (4.22), which represents the mobile robot's relative position, which is then finally passed back into v, equation (4.28), and new instructions are sent to the mobile robot if they do not match the desired path.

4.5 PD Controller Simulation Results

Using the simulation tools available in MATLAB an experiment was set up to explore the behaviour of the PD controller, presented in Figure 4-4.

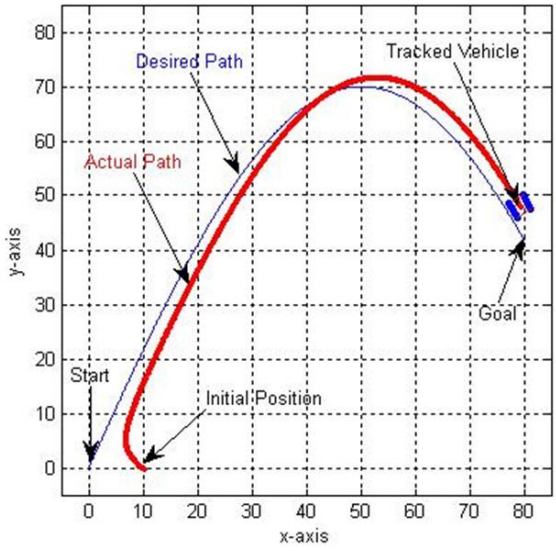


Figure 4-5 MATLAB Path Following Experiment With PD Controller

As presented in Figure 4-5, the experiment was set up with a starting point and a goal point. The mobile robot was set up with an offset pose and location different to the targeted starting point. The observed behaviour from Figure 4-5 showed how the mobile robot was using the PD controller and linear feedback to follow a target polynomial path. Based on the simulation, the observation was that the PD controller behaved as expected.

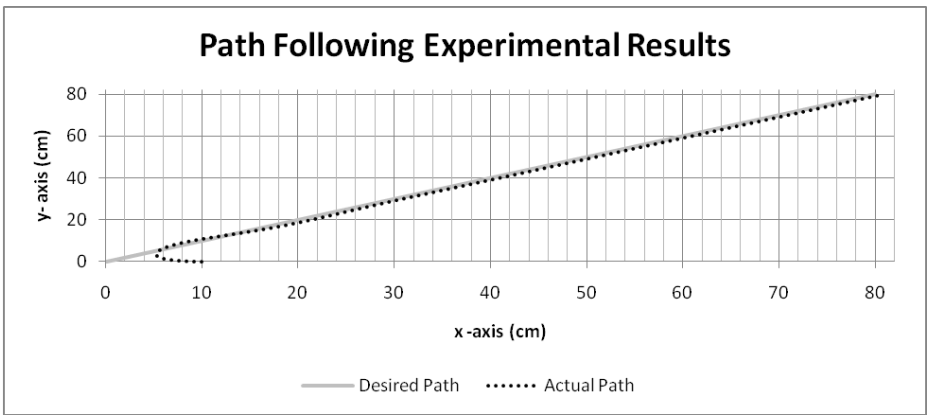


Figure 4-6 Linear Path Following Simulation

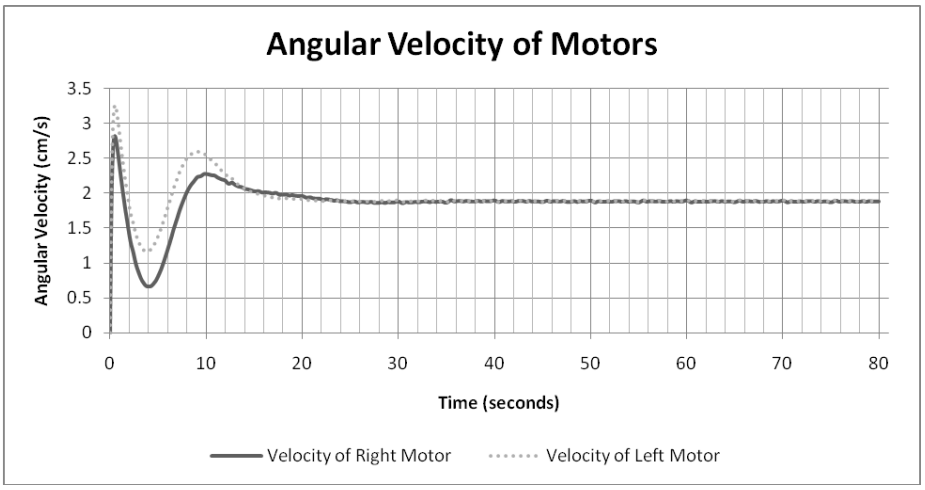


Figure 4-7 Feedback Angular Velocity from Linear Path Following Simulation

Using a simpler linear path to follow, a second experiment was set up to observe the behaviour of the PD controller. The results shown in Figure 4-6 present the simulated behaviour of the mobile robot following a linear line. The output behaviour of the motors is also presented in Figure 4-7, showing the smooth transition effects of the Lagrange multipliers and PD controller constants used to optimize the behaviour of the mobile robot. The observation based on the results presented here was that the PD controller was behaving as expected.

4.6 Summary & Conclusions

This chapter covered the research carried out in defining the constraints, the model of the system, and an optimized controller for a manoeuvrable non-holonomic two-wheel mobile robot. The following research outcomes were presented in this chapter:

- The holonomic and nonholonomic constraints and singularities of a two-wheel manoeuvrable mobile robot were derived.
- Using the constraints and singularities the equations of motion for the mobile robot were derived using the Lagrangian of the system.
- Using the equations of motion the system was optimized using Lagrange multipliers on the system constraints.
- Using the optimized model of the mobile robot, a closed-loop PD controller with linear feedback was derived.
- Validation of the PD controller was carried out using simulated experiments.

The research delivered in this chapter provides a closed-loop system optimized PD controller for a two-wheel manoeuvrable mobile robot. Researchers carrying out early research in two-wheel manoeuvrable mobile robots can apply this optimized controller to their mobile robots.

The research here is limited to the configuration of a two-wheel manoeuvrable mobile robot that has nonholonomic constraints. The derived equations of motion assume that there is no slip generated by the wheels and, as such, this is not considered. It also assumed that the centre of gravity for the system is not offset from the centre point of mobile robot. For the PD controller, the proportional and derivative gains will be different for every mobile robot and every system will need to tune these constants before using the controller. The simulation results derived from the experiment are based on simulations only.

The simulated results validated the optimized behaviour of the closed-loop PD controller, which showed how the mobile robot closely followed the desired path. While the results were favourable, further research is still possible in exploring how this system would behave with a hybrid with an open-loop controller. The ideal scenario is for the closed-loop controller to wait

for feedback from sensors before performing the next action while the open-loop controller carries out the corrective action before something occurs that would prevent a correction. It is possible that this preventive approach can only further enhance the control system for this type of mobile robot configuration.

Chapter 5 Self-localization using Single & Double Compass Configuration

5.1 Introduction

In the previous chapter, Chapter 4, research on the nonholonomic constraints and the kinematic and dynamic model of the two-wheel manoeuvrable mobile robot were presented. The research also presented an optimized closed-loop PD controller which is dependent on linear feedback. The feedback into the controller is critical to the behaviour of the mobile robot, for instance if the feedback is not accurate, the resulting outcome from the controller will be erroneous and impact where and how quickly the mobile robot gets to its targets. Bearing in mind that the sensor feedback is so critical to the behaviour of the mobile robot, this chapter presents the research carried out using the mobile robot's on-board sensors to feedback the mobile robot's orientation and position in a two dimensional space.

The term sensor feedback for this research problem can be effectively described as self-localization, as the data from the sensors is transformed into the localization variables which are fed into the derived PD controller from the previous chapter. The motivation for the research in this chapter is that the self-localization topic is still an active research topic and has not returned an absolute solution to the problem. Research into analytical solutions that resolve the orientation and position of the mobile robot will benefit the mobile robot community and will assist in providing better navigation performance in mobile robots.

For the PD controller to perform at its optimum, the feedback into the controller has to be accurate. The feedback into the PD controller includes the mobile robot's current orientation and position and, as such, this is better described as a self-localization problem. There are a couple of research methods available to solve this problem, for such as the research by Guo et al [59] on exploiting the Monte Carlo Localization (MCL) algorithm which requires detailed information about the navigation environment to provide an estimate of localization. An alternative to the Monte Carlo method is the Markov method as presented by Fox et al [60] and, again in this method, detailed information about the environment is needed. However, what is the solution when your mobile robot does not have detailed information about the environment?

Since the mobile robot has quadrature shaft encoders, it is possible to resolve localization of the mobile robot as suggested by the research presented by Lee et al [61] which uses an optical flow sensor and the wheel encoders. But this method is flawed just like any approach that is numerical because it will lead to an accumulation of errors over time, rendering the result erroneous. It is possible to use an alternative sensor, rather than the wheel encoders, like the research presented by North et al [62] which presents a system that exploits GPS localization systems. Again, even though the research is a contribution, it does not solve the problem for mobile robots that work indoors where satellites are not observable, and also the challenge for the GPS system is that it is not capable of returning accurate results for small robots. By and large, the research here proposes a more elegant solution to the numerical approach by presenting an analytical solution to the problem. To achieve an analytical solution, the following objectives were set:

- Derive the electrical inputs that drive the two-wheel manoeuvrable mobile robot.
- Derive the model for translating the shaft encoder output to orientation and positional coordinates for the mobile robot.
- Derive the model for using a single compass to resolve a numerical solution to the mobile robot location.
- Derive the model for using a double compass to resolve an analytical solution to the mobile robots location.
- Validate the novel double compass analytical solution using an overhead computer vision tracking system.

Resolving an analytical solution, as opposed to a numerical response that accumulates errors is an important research topic, which can benefit the mobile robot community.

The research in the chapter is specific to two-wheel manoeuvrable mobile robots with quadrature shaft encoders mounted on the wheel motors and digital orientation compasses positioned directly above the wheels of the mobile robot. When considering the model for computing the travelled distance of each wheel, it is assumed that the wheel does not suffer from slip. When considering the acquisition of orientation from the digital compass sensors, it is assumed that the sensor is calibrated and does not suffer from hard or soft iron interference.

5.2 A Hybrid Dual Shaft Encoder Configuration for Localization

The manoeuvrable mobile robot is configured with two independent direct current (DC) servo motors, set up to be parallel to each other, with the ability for continuous rotation. This configuration allows the mobile robot to behave in a manoeuvrable configuration, and is illustrated below:

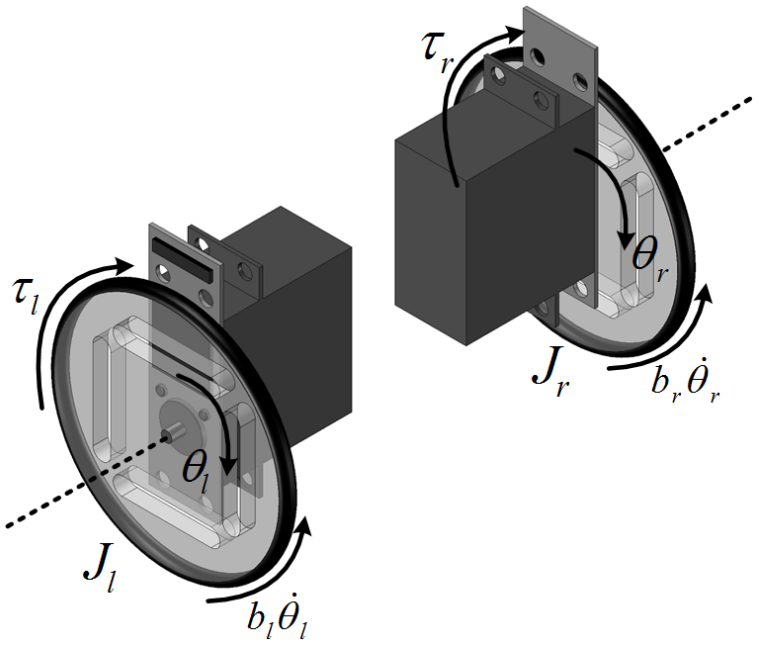


Figure 5-1 Dual Servo Motor Drive Configuration

It is assumed that both the left and right servo motors are identical. The torque, $\tau_{l,r}$, of the motor is related to the armature current, $i_{l,r}$, by the constant factor K_t and is described as $\tau_{l,r} = K_t.i_{l,r}$. The input voltage source is described as $V_{l,r}$, which is used to drive the servo motors, where $R_{l,r}$ is internal resistance of the motor, where $L_{l,r}$ is the internal inductance of the motor, and where $e_{l,r}$ describes back electromagnetic field (EMF) of both the left and right electric servo motors. It is known that $e_{l,r} = K.\dot{\theta}_{l,r}$, where $K = K_e = K_t$ describe the electromotive force constants.

Symbol	Description
$I_{wl,r}$	Moment of inertia of the rotor (kg.m²)
$b_{l,r}$	Damping ratio of the mechanical system (Nms)
K	Electromotive force constant (V/rad/sec)
$R_{l,r}$	Electric resistance (ohm)
$L_{l,r}$	Electric inductance (H)
$V_{l,r}$	Input voltage source (V)

Table 5-1 Dual Servo Motor Configuration Definitions

The value definitions listed in Table 5-1 complements Figure 5-1, which represents the mobile robot drive configuration.

Using Newton's laws of motion and Kirchhoff's circuit laws [63] and [64], the motion of the motors can be related to the electrical behaviour of the circuit.

$$I_{wl,r}\ddot{\theta}_{l,r} + b_{l,r}\dot{\theta}_{l,r} = Ki_{l,r} \tag{5.1}$$

$$L_{l,r}\frac{di_{l,r}}{dt} + R_{l,r}i_{l,r} = V_{l,r} - K\dot{\theta}_{l,r} \tag{5.2}$$

Where equation (5.1) describes the Newtonian derivation of motion of both motors and where equation (5.2) describes how the circuit behaves applying to Kirchhoff's laws. Having derived equations (5.1) and (5.2), the next step would be to relate the electrical circuit behaviour to the mechanical behaviour of rotating motors, which is achieved using Laplace transforms and expressing equations (5.1) and (5.2) in terms of s as follows:

$$s(I_{wl,r}s + b_{l,r})\theta_{l,r}(s) = KI_{l,r}(s) \tag{5.3}$$

$$(L_{l,r}s + R_{l,r})I_{l,r}(s) = V_{l,r} - Ks\theta_{l,r}(s) \tag{5.4}$$

Using equations (5.3) and (5.4), the open-loop transfer function of this configuration can be derived by eliminating $I_{l,r}(s)$ and relating the equation of motion to the circuit behaviour as follows:

$$\frac{\dot{\theta}}{V_{l,r}} = \frac{K}{(I_{wl,r}s + b_{l,r})(L_{l,r}s + R_{l,r}) + K^2} \quad \frac{rad/\sec}{V} \tag{5.5}$$

Where equation (5.5) equates the rotational speed of the motors as the system's output and the voltage applied to the motors as the system's input.

By using the quadrature shaft encoders that accumulate the distance travelled by the wheels, a form of position can be deduced by deriving the mobile robot's x, y Cartesian position and the manoeuvrable vehicle's orientation ϕ, with respect to time. The derivation starts by defining and considering, $v_s(t)$, which is the velocity as a function of time for the mobile robot, and $\phi(\mathrm{t})$ is the orientation of the mobile robot as a function of time. The velocity and orientation are derived from differentiating the position form as follows:

$$\frac{dx}{dt} = v_s(t)\cos(\phi(t)) \tag{5.6}$$

$$\frac{dy}{dt} = v_s(t)\sin(\phi(t)) \tag{5.7}$$

The change in orientation with respect to time is the angular velocity ω, which was defined in equation (4.4) and can be described as follows:

$$\frac{d\phi}{dt} = \omega = \frac{1}{L}\left(\frac{dv_r}{dt} - \frac{dv_l}{dt}\right) \tag{5.8}$$

When equation (5.8) is integrated, the mobile robot's angle orientation value $\phi(\mathrm{t})$ with respect to time, is achieved. Where, v_r and v_l are the translational velocity of the right and left wheels, respectively. The mobile robot's initial angle of orientation $\phi(0)$ is written as ϕ_0 and is represented as follows:

$$\phi(t) = \frac{(v_r - v_l)t}{L} + \phi_0 \tag{5.9}$$

The velocity of the mobile robot is equal to the average speed of the two wheels and this can be incorporated into equations (5.6) and (5.7), which are depicted as follows:

$$\frac{dx}{dt} = \frac{v_r + v_l}{2}\cos(\phi(t)) \tag{5.10}$$

$$\frac{dy}{dt} = \frac{v_r + v_l}{2}\sin(\phi(t)) \tag{5.11}$$

The next step is to integrate equations (5.10) and (5.11) to the initial position of the mobile robot, which is depicted as follows:

$$x(t) = x_0 + \frac{L(v_r + v_l)}{2(v_r - v_l)}\left(\sin\left(\frac{(v_r - v_l)t}{L} + \phi_0\right) - \sin(\phi_0)\right) \tag{5.12}$$

$$y(t) = y_0 + \frac{L(v_r + v_l)}{2(v_r - v_l)}\left(\cos\left(\frac{(v_r - v_l)t}{L} + \phi_0\right) - \cos(\phi_0)\right) \tag{5.13}$$

Equations (5.12) and (5.13) describe the mobile robot's position, where $\mathrm{x}(0) = \mathrm{x}_0$ and $\mathrm{y}(0) = \mathrm{y}_0$ are the mobile robot's initial positions. The next step is to represent equations (5.9), (5.12) and (5.13) in terms of the distances that the left and right wheels have traversed, which are defined by d_R and d_L. This can be achieved by substituting $\dot{\theta}_\mathrm{r}$ and $\dot{\theta}_\mathrm{l}$ (in equations (5.9), (5.12) and (5.13)) for d_R and d_L, respectively, and also dropping the time constant *t* to achieve the following:

$$\theta = \frac{d_R - d_L}{2L} + \theta_0 \tag{5.14}$$

$$x(t) = x_0 + \frac{L(d_R + d_L)}{2(d_R - d_L)}\left(\sin\left(\frac{(d_R - d_L)}{L} + \phi_0\right) - \sin(\phi_0)\right) \tag{5.15}$$

$$y(t) = y_0 + \frac{L(d_R + d_L)}{2(d_R - d_L)}\left(\cos\left(\frac{(d_R - d_L)}{L} + \phi_0\right) - \cos(\phi_0)\right) \tag{5.16}$$

By implementing equations (5.14) to (5.16), they provide a solution to the relative position of a manoeuvrable mobile robot. This might offer a possible solution to the self-localization problem but is subject to accumulative drift of the position and orientation with no method of re-alignment. The accuracy of this method is subject to the sampling rate of the data accumulation,

such that if small position or orientation changes are not recorded, then the position and orientation will be erroneous.

5.3 A Numerical Model for Localization with a Digital Compass

Having derived a self-localization model using only the telemetry from the quadrature shaft encoders, the next step to evolve the model is to add a digital compass, such as a magnetometer [65], to input the manoeuvrable mobile robot's bearing with respect to magnetic north. The overall objective is to use the mobile robots bearing angle, from the magnetometer, and the feedback from the wheel shaft encoders to deduce an accurate orientation and location of the mobile robot, to ultimately give the mobile robot platform the ability of self-localization. The ideal position on the vehicle for the digital compass would be at the midpoint between the centre of its mass and the intersection of the axis of symmetry with the mobile robot's driving wheel axis. In this case, the vehicle is configured such that there is no deviation between the two points, P_o and P_c.

When the manoeuvrable mobile robot starts a forward or backward rotation configuration, it induces two independent instantaneous centres of curvatures from the left and right wheel.

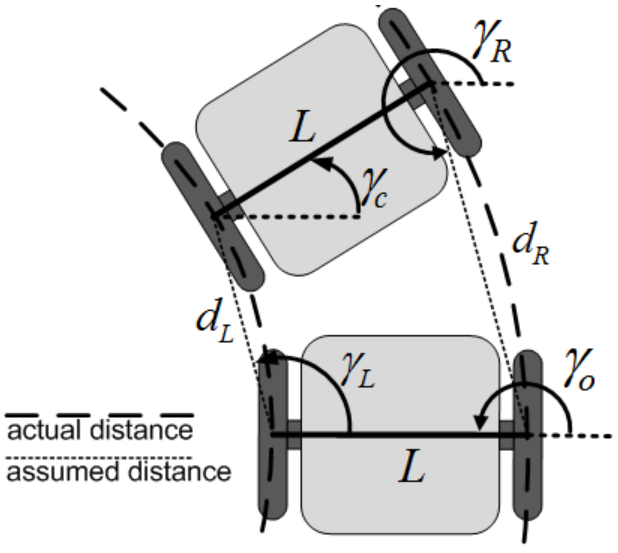


Figure 5-2 Mobile Robot Manoeuvre Configuration

Using a steady state manoeuvre, it is assumed that the actual distance travelled in a rotation manoeuvre does not equal the distance used to model the calculation for position and orientation. This assumption is depicted in Figure 5-2, clearly showing the actual and assumed

distance travelled. The difference does not cause any consequence to the calculation model as the difference cannot be measured in the resolution of the quadrature shaft encoders.

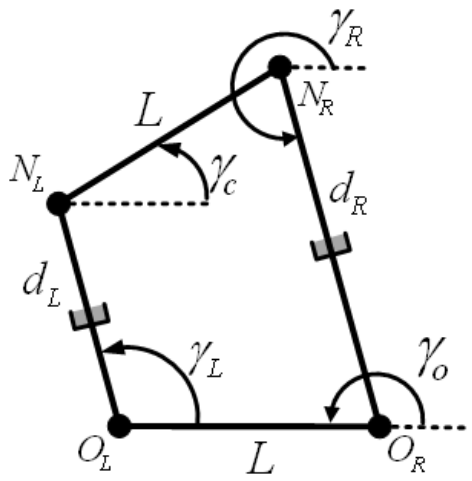


Figure 5-3 Six-bar Linkage Model for a Single Compass Configuration

Using the vector loop technique [66] to analyse the kinematic position of the linkage model in Figure 5-3, the vector loop equation is written as follows:

$$I_{N_L O_L} + I_{N_R N_L} + I_{O_R N_R} + I_{O_L O_R} = 0 \tag{5.17}$$

Using the complex notation, equation (5.17) is written as follows:

$$d_L.e^{j.\gamma_L} + L.e^{j.\gamma_c} + d_r.e^{j.\gamma_R} + L.e^{j.\gamma_o} = 0 \tag{5.18}$$

Having derived the complex notation of the vector loop equation in equation (5.18), the next step is to substitute the Euler equations to the complex notations as follows:

$$\begin{aligned} &d_L(\cos(\gamma_L) + j\sin(\gamma_L)) + L(\cos(\gamma_c) + j\sin(\gamma_c)) + \\ &d_r(\cos(\gamma_R) + j\sin(\gamma_R)) + L(\cos(\gamma_o) + j\sin(\gamma_o)) = 0 \end{aligned} \tag{5.19}$$

Equation (5.19) is separated into its corresponding real and imaginary parts, considering $\gamma_o = 180°$, as follows:

$$d_L \cos(\gamma_L) + L\cos(\gamma_c) + d_r \cos(\gamma_R) - L = 0 \tag{5.20}$$

$$d_L \sin(\gamma_L) + L\sin(\gamma_c) + d_r \sin(\gamma_R) = 0 \tag{5.21}$$

where, d_L, d_R, and L are all known constants, and γ_c specifies the new angle of orientation of the mobile robot. Having two equations (5.20) and (5.21), with two unknown angles γ_L and γ_R, when solving simultaneously for these two angles, it will return multiple independent values. This approach requires a large amount of computation to first find these angles and then deduce the relative position.

5.4 An Analytical Model for Localization with a Dual Digital Bearing Compass

Having described the issues associated with using a single compass to solve for the position above, it is clear that it would be preferable to have a model that eliminated having simultaneous solutions and led to a single solution. This would ideally mean that the angles γ_L and γ_R are known constants, and to achieve this condition would require precision telemetry from the digital compass. A model with a compass is proposed to resolve the angles γ_L and γ_R, as shown in Figure 5-3.

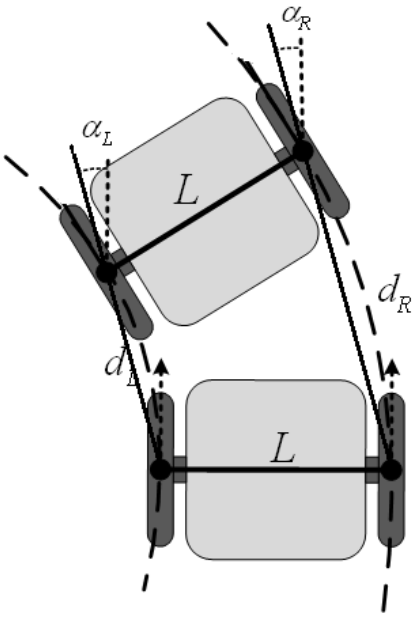


Figure 5-4 Double Compass Manoeuvre Configuration

To enable higher precision, a configuration of two compasses is introduced, which are placed directly above the rotating wheels when a configuration change takes place, the difference is measured by α_L and α_R, which represent the change in orientation of the left and right mobile robot wheels, respectively.

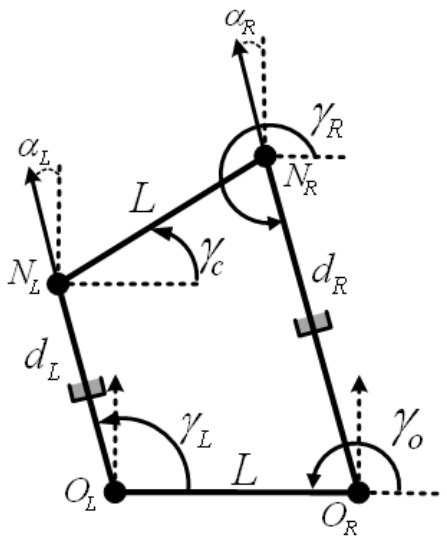


Figure 5-5 Six-bar Linkage Manoeuvre Model with a Double Compass Configuration

Using the same approach as a single compass configuration, the double compass manoeuvre configuration is modelled using a six-bar mechanism as shown in Figure 5-5.

Using an identical approach as the single compass configuration, the vector loop equations remain the same, equations (5.20) and (5.21), with the difference being the ability to define the angles γ_L and γ_R, and using the mean value of the two compasses for higher precision. Where the mean bearing from the two compasses is defined as follows:

$$a = \frac{a_L + a_R}{2} \tag{5.22}$$

For the manoeuvre model, presented in Figure 5-5, using the mean bearing angle from equation (5.22), the configuration can be calculated as follows:

$$\gamma_L = \beta_L + a \tag{5.23}$$

$$\gamma_R = \beta_R + a \tag{5.24}$$

where, β_L and β_R are the trigonometric angles used for calculating γ_L and γ_R for each pose of the mobile robot, based on the different configuration states of d_L and d_R. For the configuration state described by Figure 5-4, $\beta_L = 90°$ and $\beta_R = 270°$. Having a constant value

for the angles γ_L and γ_R from Equations (5.23) and (5.24), it allows either equation (5.20) or (5.21) to be used to derive the remaining angle γ_c, which is specified as follows:

$$\gamma_c = \cos^{-1}\left(\frac{-d_L \cos(\gamma_L) - d_r \cos(\gamma_R) + L}{L}\right) \tag{5.25}$$

$$\gamma_c = \sin^{-1}\left(\frac{-d_L \sin(\gamma_L) - d_r \sin(\gamma_R)}{L}\right) \tag{5.26}$$

where equations (5.25) and (5.26) solve the single compass solution to the position of the centrally positioned digital compass, indicated by γ_c.

Using the simultaneous solutions from comparing equations (5.20) and (5.21), and evaluating their difference to either equation (5.25) or (5.26), it is possible to derive the accuracy of the measurement. The simultaneous equation solution will be the best fit model for the hybrid model, and any ambiguity can be considered resolution error or a factor of wheel slippage.

5.5 A Comparing a Numeric Model against an Analytical Dual Compass Model

To validate the double compass approach for self-localisation of the mobile robot, an overhead camera visual odometry tracking system was set up, which is presented in more detail in Chapter 7. Visual tracking results are used to compare the double compass tracking results and statistical analysis is carried out The results are as follows.

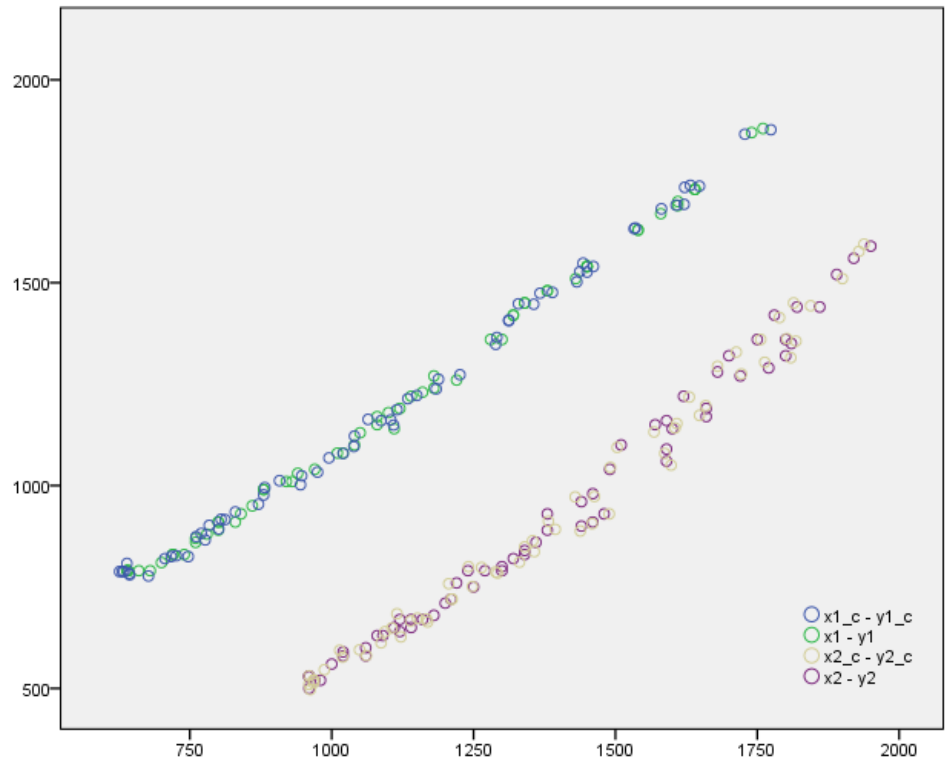


Figure 5-6 Experimental data from linear path following

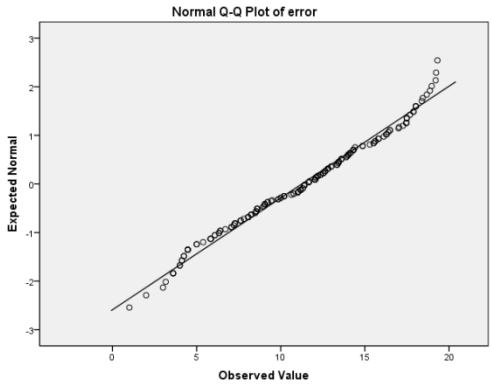
Figure 5-7 Experimental data from sinusoidal path following

Using the configuration presented in Figure 7-3, which exploits the use of an overhead camera system with visual markers to observe the mobile robots position and orientation. The mobile robot with a double compass configuration was set up to follow a linear path and a sinusoidal path. The experimental results showing the visual tracking results and the double compass estimation results are presented in Figure 5-6 and Figure 5-7. The position data from left and right mobile robot wheel are denoted as x1-y1 and x2-y2, respectively, and the position data from the left and right mobile robot markers is denoted as x1_c-y1_c and x2_c-y2_c. Visually inspecting the data presented in Figure 5-6 and Figure 5-7, it can be observed that both sets of data show similar results. While, the data in Figure 5-6 shows a near linear behaviour, the error and devation seems to suggest some slip and the additional error can be attributed to resolution and tollereance of the sensors. The error from slipping is more noticable in Figure 5-7, but the sinasodal behaviour is noticable by visual inspection.

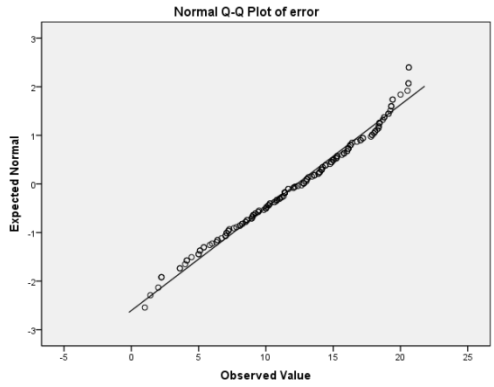
Method	Right Marker	Left Marker
Mean	12.2983	11.2544
95% Confidence Interval for Mean (LB)	11.6059	10.6166
95% Confidence Interval for Mean (UB)	12.9906	11.8922
Median	12.8062	11.4017
Variance	22.2840	18.9110
Standard Deviation	4.7205	4.3487
Standard Error Mean	0.3509	0.3232
Minimum	1.0000	1.0000
Maximum	20.6155	19.3132
Range	19.6155	18.3132
Interquartile Range	7.0312	6.2556
Skewness	-0.2630	-0.1180

Table 5-2 Statistical analysis of double compass experimental data

The results presented in Table 3 show the statistical analyses of 362 samples recorded from the linear and sinusoidal manoeuvre experiments. Both left and right markers mean values are relatively low and, for more accuracy, because the data might be skewed, the median value is presented because it compensates for skewed data. The confidence intervals, which represent two standard deviations from the mean, equivalently present a low error rate.



Left Marker



Right Marker

Figure 5-8 Normal Q-Q Plot of Error for the Right and Left Markers

The Q-Q plot depicted in Figure 5-8 presents the performance of the observed values against the expected values. The Q-Q plot is the plot of the quantiles of the mobile robots self-localisation method for orientation data against the quantiles of the overhead camera marker tracking system data. If the data sets come from a population with the same distribution, the points should fall approximately along the reference line. The greater the departure from the reference line, the greater the evidence for the conclusion that the two data sets have come from populations with different distributions. The data from the Q-Q plot demonstrates that the errors are approximately normally distributed centrally, with anomalies at both tail ends, which infers that the data from the mobile robots self-localisation method for orientation and the results from the overhead camera marker tracking system come from the same population with the same distribution.

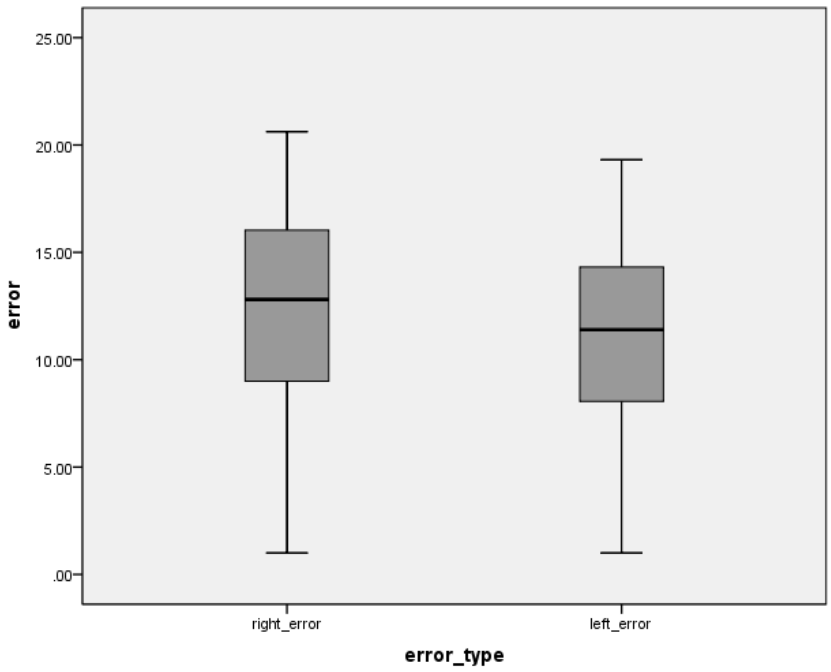


Figure 5-9 Boxplot of Double Compass Errors base on the Experimental Methods

The boxplot presented in Figure 5-9 visually shows the distance from 0 to the mean, which is 12.3mm and 11.3mm, for the right and left mobile robot wheel point orientation mean errors, respectively. It also presents the interquartile range, which is 19.6mm and 18.3mm, for the right and left mobile robot wheel point orientation mean errors, respectively, and minimum (1mm) and the maximum (20.6mm and 19.3mm) values. The boxplot is a useful plot showing the mean error, the expected range using interquartile range of the mean errors, and the maximum and minimum mean error.

Using the statistical analysis presented in the previous section, a non-parametric test is required to validate the practical effectiveness of the double compass methodology. The ideal analysis test for a non-parametric independent one-sample set of data is the Kolmogorov-Smirnov test [67] for significance. The empirical distribution function F_n for n independent and identically distributed random variables observations, X_i is defined as follows:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{X_i \le x} \tag{5.27}$$

where $I_{X_i \le x}$ describes the indicator function, which is equal to 1 if $X_i \le x$ and is equal to 0 otherwise. The Kolmogorov-Smirnov statistic [67] for a cumulative distribution function $F(x)$ is as follows:

$$D_n = \sup_x |F_n(x) - F(x)| \tag{5.28}$$

where $\sup_x$ describes the supremum of the set of distances. For the analysis test to be effective to reject a null hypothesis, a relatively large number of data is required.

Under the null hypothesis that the sample originates from the hypothesized distribution $F(x)$, the distribution is specified as follows:

$$\sqrt{n} D_n \xrightarrow{n \to \infty} \sup_t |B(F(t))| \tag{5.29}$$

where $B(t)$ describes the Brownian bridge [68]. If F is continuous, then under the null hypothesis $\sqrt{n} D_n$ converges to the Kolmogorov distribution, which does not depend on F. The analysis test is constructed by using the critical values of the Kolmogorov distribution. The null hypothesis is rejected at level α if $\sqrt{n} D_n > K_a$, where K_a is calculated from the following:

$$\Pr(K \le K_a) = 1 - \alpha \tag{5.30}$$

It should also be noted that the asymptotic power of this analysis test is 1.

For the experimental data presented in this paper, the null hypothesis is that the distribution of error is normal with a mean value of 11.78 and a standard deviation of 4.56. Based on the significance level of 0.05, a significance of 0.663 is returned using the one-sample Kolmogorov-Smirnov test (5.30). The strength of the returned significance value allows us to retain the null hypothesis and say that the distribution of error is normal, with a mean value of 11.78 and a standard deviation of 0.663.

5.6 Summary & Conclusions

This chapter provided a novel analytical approach to solving self-localization using a hybrid configuration with a mobile robot's quadrature wheel encoder and digital compasses. The following research outcomes were presented in this chapter:

- An electrical model required to manage power distribution to motors is provided.
- The heuristic formula for computing the mobile robot's orientation and position, using only quadrature shafts, is derived.
- The numerical based model, using a single compass with the quadrature encoders, is derived based on a 6-bar mechanism.
- The analytical based model using a double compass configuration with the quadrature encoders is derived based on the 6-bar mechanism.
- A statistical analysis with a null hypothesis is validated to show the self-localization results from the analytical method against an overhead tracking system.

The research covered in this chapter delivered a novel approach for resolving self-localization in two-wheel manoeuvrable mobile robots. The novel approach provides an analytical, rather than numerical solution, which does not suffer from an accumulating error, common in numerical methods. Also, the analytical methods are known to be less computational than numerical methods, which is a benefit to small mobile robots with diminished computation power on-board. The results compared to the computer vision overhead tracking system showed favourable results with a mean error of less than 15mm. Thus, the objective to provide accurate feedback to the optimized PD controller, from Chapter 4, has been achieved with this derived method.

The novel research into the double compass configuration with the analytical solution is a valuable contribution to the mobile robot community, providing a self-localization method for small mobile robots with constrained computation power and for indoor mobile robots without the capabilities of environment map building and GPS.

The statistical results presented favourable results compared to the overhead camera tracking system. With a mean error of 12mm, this self-localization method will make a valuable contribution to mobile robots that require self-localization feedback for path planning controllers.

The research provided an analytical solution to the self-localization problem which is bespoken and applicable only to a two-wheel manoeuvrable mobile robot configured with quadrature wheel encoders and a dual configuration of digital compasses. The results of this method are limited by the accuracy of the magnetometers used to calculate orientation. If the magnetometer is un-calibrated, and if the sensor is affected by noise from hard and soft irons, the results will be erroneous. It is also important to point out the limitations of the quadrature shaft encoders, as this is a mechanical sensor, it has a finite resolution. If low-resolution wheel encoders are used and the mobile robot makes very small movements, the sensor feedback will not be measured, which will lead to erroneous results. It is also important to note that this model assumes that the wheels for the mobile robot are free from slip.

For the results of the analytical solution to be valid, it is assumed that the mobile robot's wheels are slip free. To build a more robust solution, a model for slip estimation needs to be incorporated to eliminate the chance of erroneous results because of wheel slip. The research by Balakrishna et al [69] provides a model for estimating slip in wheeled mobile robots that can be explored further.

Chapter 6 Self-localization using a 9-axis IMU Sensor with a Directional Cosine Matrix

6.1 Introduction

In Chapter 4, the research into an optimized PD controller was derived which was designed to facilitate the behaviour of the mobile robot moving from point A to point B. Then, in Chapter 5, the research into preparing the self-localization feedback for the PD controller was presented with a novel analytical self-localization method that uses the mobile robot's wheel encoders and a dual magnetometer configuration. While the addition of the more accurate feedback improved the overall system, both chapters also highlighted a significant limitation of the control model and the self-localization feedback method. Neither the controller model nor the self-localization feedback method took slip estimation into consideration. In addition to the limitation of overlooking slip estimation, the two-wheel drive system consumes a significant amount of power from the battery cells, and research into improving the distribution of power to the motors is required. The research of Yong et al [70] presents a model for two-wheel self-balancing mobile robots similar in design to that of the KCLBOT but without the caster wheels for balance. The research presented a Takagi-Sugeno fuzzy controller model [71] to balance the mobile robot, with a parallel distribution compensator (PDC) to optimize the distribution to the motors. Further to the model by Yong et al, the research by Hao et al [72], on a two-wheel self-balancing motor cycle, presented an analysis of how the coupling dynamic system and control system can lead to better efficiency and reliability in two-wheel self-balancing systems. The research in this chapter will build on these research ideas and continue on the theme of self-localization along with the novel approach taken, using the fusion of absolute sensors to form an inertial measurement unit to solve the self-localization problem.

When the two-wheel manoeuvrable mobile robot system carries out any type of motion, forward or backward, power is driven to the motors, which then rotates the wheels. As a result of this initial power from the motors, the frame of the mobile robot is rotated with wheels until the caster wheels are fully depressed. After this motion, where the caster wheels are exploited to balance the mobile robot, it comes with the consequence that this action will cause the wheels to slip. The ideal solution is to prevent the caster wheels from being fully depressed. To achieve

this, the mobile robot system requires feedback of its orientation in a three dimensional space. The ideal solution to achieve this with an inertial measurement unit is as follows:

- Derive the configuration of the inertial measurement unit using an accelerometer, magnetometer, and a gyroscope to form a 9-axis sensor.
- Derive the system's frame of reference and the rotation matrix with orthogonal constraints used as the directional cosine matrix.
- Solve the systems pose using the system's frame and the directional cosine matrix.
- Implement a closed-loop proportional/integral PI controller to correct errors evolving from the sensor and optimize the output with fixed gains.
- Validate the optimized PI controller using a 3D computer vision tracking system.

The optimized PI controller will provide accurate three-dimensional orientation localization feedback that ultimately can be used to configure the mobile robot at a self-balancing system to prevent slipping from occurring on the wheels.

The accelerometer is biased towards gravity and is ideal for measuring orientation when the device is not moving quickly and is free of vibrations. Under these conditions, the sensor returns accurate measurements of roll and pitch. The magnetometer is biased towards the Earth's magnetic north pole and measures the bearing angle of the device, but it is subject to interference from hard and soft irons. The magnetometer measurement needs to be compensated if the device is tilted, as the measurement will be inaccurate if used uncompensated. The gyroscope returns the most accurate measurement of orientation of roll, pitch and yaw. This is achieved by the sensor measuring the rate of change of rotation about these angles. To return the positional measurement, the values are subjected to integration, which leads to the sensor value being subject to accumulative drift after time, so the sensor values are only accurate for a short period of time.

6.2 A 9-axis IMU Sensor

For the IMU to be effective, the different measurement sensors need to be placed close to each other and they need their axis orientations to be aligned. Analog Devices ADXL345 3-axis accelerometer with 13-bit measurements at up to ±16g, the Honeywell HMC5883L 3-axis magnetometer with 12-bit measurements which achieves 2-milli gauss resolution in ±8 gauss fields, and the InvenSense ITG-3200 3-axis gyroscope with 16-bit measurements with a sensitivity of 14 LSBs per º/sec and a full-scale range of ±2000 º/sec, were selected for the sensor setup.

6.2.1 Inertial Measurement Unit (IMU) Hardware Setup

For the IMU to be effective, the different measurement sensors need to be placed close to each other and need their axis orientations aligned.

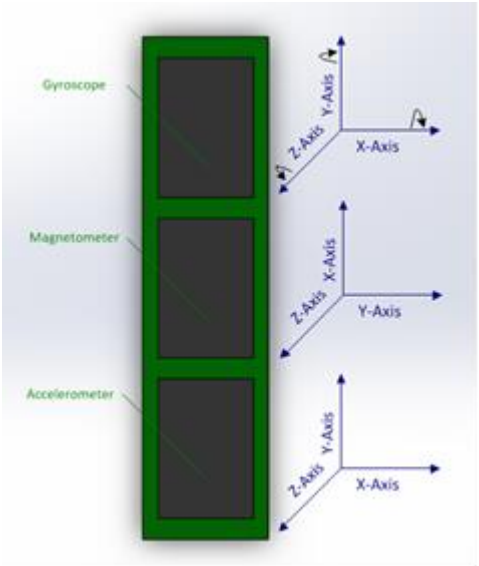


Figure 6-1 Inertial Measurement Unit (IMU) Sensor Setup

In Figure 6-1, we show how the sensor is set up, how the three sensors are orientated against each other and how their planes are aligned with each other.

$$g = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \tag{6.1}$$

The column vector g from equation (6.1) holds the 3-axis measurements from the gyroscope sensor. Before these measurements can be used, the values for the sensor need to be offset, which is done by placing the sensor on a level, stationary surface and capturing a large sample of measurement. The measurements are averaged and then the average value is subtracted from the measurement value in equation (6.1). If the average value is good when the sensor is stationary, the measurements returned should be zero.

$$a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{6.2}$$

The column vector a from equation (6.2) holds the 3-axis measurements from the accelerometer sensor. Before the measurement from the accelerometer can be used, the measurements need to be offset, which is done by calibration, and multiplied by a scaling factor, which is provided by the manufacturer of the sensor.

$$m = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{6.3}$$

The column vector m from equation (6.3) holds the 3-axis measurements from the magnetometer sensor. The magnetometer will return very noisy measurements if it is not calibrated. Like the accelerometer, the measurement needs to be offset and then multiplied by a scaling factor. The offset is calculated from the calibration and the scaling factor is provided by the manufacturer. To deal with the noise from hard and soft irons, an ellipsoid fitting method [73] is used in the enhancement of the accuracy of the measurement values from the magnetometer sensor.

After the magnetometer measurement m has been captured from equation (6.3), the next step is to use these measurements to observe the heading of the sensor, which is denoted as ψ as the heading angle and the rotation about the z axis. Because the sensor is not perpendicular to the plane of the earth, if the measurements are used as they are now, the heading angle will be inaccurate. To improve the accuracy of the heading angle, the tilt compensation [74] is done by

using the Euler angles returned for pitch and roll, which are expressed as θ and ϕ, respectively.

$$\psi = \arctan\left(\frac{-\left(m_x \cos(\theta) + m_y \sin(\phi)\sin(\theta) + m_z \cos(\phi)\sin(\theta)\right)}{m_y \cos(\phi) - m_z \sin(\phi)}\right) \tag{6.4}$$

where the numerator in the arc tangent in equation (6.4) describes the magnetic forces in the x axis and the denominator describes the magnetic forces in the y axis.

6.3 Directional Cosine Matrix for Self-Localization

The idea is to use a directional cosine based matrix that will resolve a rotation matrix to allow the derivation of a moving frame of reference to a stationary fixed frame reference.

$$q = \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \tag{6.5}$$

where q describes a column vector, which holds the orientations measurements of the moving frame of reference. The values in q describe the 3-dimensional orientation of the sensor being observed, which is moving in free space, with reference to fixed frame of reference, which is described as follows.

$$f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{6.6}$$

where f describes a column vector, which holds the orientation measurements of the fixed frame of reference that is used as point of reference to solve the orientation of the q column vector.

$$f = Rq \tag{6.7}$$

The idea is to use a 3x3 rotation matrix R to estimate the pose of q from equation (6.5). Since f is fixed frame of reference and is not variable, as described in equation (6.6), the rotation matrix [75] is described as follows:

$$R = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} \tag{6.8}$$

Since we are interested in the Euler angles, the rotation matrix R from equation (6.8) is better described as follows:

$$R = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad (6.9)$$

where (θ, ϕ, ψ) from equation (6.9) is derived from Figure 6-1. Pitch, described by θ, is the rotation about the y axis; roll, described by ϕ, is the rotation about the x axis; and yaw, described by ψ, is the rotation about the z axis.

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \quad (6.10)$$

Substituting equation (6.7) with equations (6.5), (6.6), and (6.8) derives equation (6.10) where we have a 3x3 matrix multiplied by a 3x1 matrix, which results in the following:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} r_{xx}q_x + r_{xy}q_y + r_{xz}q_z \\ r_{yx}q_x + r_{yy}q_y + r_{yz}q_z \\ r_{zx}q_x + r_{zy}q_y + r_{zz}q_z \end{bmatrix} \quad (6.11)$$

where the transformation, derived in equation (6.11) by multiplying the rotation matrix R, resolves the vector of the fixed frame of reference. The unit values in the rotation matrix R contain the information to go from the moving frame q to the fixed frame f. We are also able to call this rotation matrix R a directional cosine matrix [76] because the entries describe the cosine angles between the fixed axis frame and the moving axis frame.

$$q = R^T f \quad (6.12)$$

Since the values of column vector f from equation (6.7) are known, and by using the transpose of the rotation matrix R from equation (6.8), we are able to solve the column vector q from equation (6.5).

The general idea is to solve for the values in the rotation matrix, described in equation (6.9), and use the fixed frame which is known constant, described in equation (6.6), to solve the pose of the sensor. To achieve this it is important to understand the behaviour for the sensors being

using and their limitations, for example the gyroscope provides the most accurate measurement of orientation. The values from the gyroscope, which are in measured as the rate of change in degrees per second, are passed into the non-linear differential kinematic equation and then normalized. Since, we are interested in the orientation position of the sensor and not the displacement over time, as such the gyroscope values are integrated at fixed time intervals to deduce this. Due to the numerical errors in the integration process, this will corrupt the orthogonal constraints of the rotation matrix R presented in equation (6.8), so corrections are required to ensure that the matrix remains orthogonal. A PI controller is used to feed in negative feedback to balance the detected errors. This is done by using a magnetometer and accelerometer, which adjusts for tilt, to detect errors in the yaw value. The accelerometer alone is used to detect errors in the pitch and roll values.

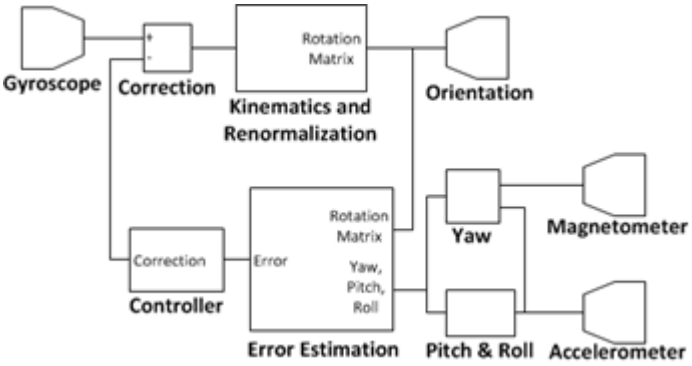


Figure 6-2 DCM Process Model

An overview of the process described above to solve the orientation of the device and compensate for accumulative errors is presented in Figure 6-2.

Before we derive the matrix that will update our rotation matrix, we need to correct the drift error. This is done by employing a PI controller, which will use a proportional and integrator term to correct the drift error.

$$g_c = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{6.13}$$

where g_c describes the corrected gyroscope measurement g from equation (6.1) after the addition of the proportional term p and integrator term n. The next step is to build a 3x3 matrix to update the existing directional cosine matrix.

$$U = \begin{bmatrix} 0 & \Delta t.g_{cz} & -\Delta t.g_{cy} \\ -\Delta t.g_{cz} & 0 & \Delta t.g_{cx} \\ \Delta t.g_{cy} & -\Delta t.g_{cx} & 0 \end{bmatrix} \tag{6.14}$$

The update matrix U from equation (6.14) is then multiplied by current directional cosine matrix R and the result is added to the current matrix.

$$R = R + RU \tag{6.15}$$

This is based on Mahony's model from his paper [77] on non-linear filters on the special orthogonal group.

Numerical errors are inherent to the system due to the method of capturing sensor measurements and because of the process described to arrive at equation (6.15). To ensure the orthogonal state of R, the process of eliminating these slowly accumulating numerical errors, called renormalization [78], is employed. The first step to renormalization is to get the first and second row from the rotation matrix R and calculate the dot product, which should be zero.

$$e = X^T Y = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \end{bmatrix} \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix} \tag{6.16}$$

where e specifies the error and the result of the error, which shows how close the X and Y rows are rotating towards each other. The dot product of X and Y is supposed to be zero.

$$X_o = \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix} - \frac{e}{2} \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix} \tag{6.17}$$

$$Y_o = \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix} - \frac{e}{2} \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix} \tag{6.18}$$

To reduce the orthogonality error, while it is known that the magnitude of each row and column of the rotation matrix (6.9) is approximately equal to one, equation (6.17) and (6.18) show how the error e is halved and rows are rotated in opposite directions by cross coupling. The result can be evaluated by substituting the new orthogonal vectors back into equation (6.16) and seeing a smaller error value. The idea is to apportion the error equally to each vector to lower the residual error after each correction, rather than assigning the correction entirely to one vector. The next step is to manage the last axis, which can be done by finding the cross product of X_o and Y_o.

$$Z_o = X_o . Y_o \tag{6.19}$$

The final step of the renormalization process is to scale all the rows of the rotation matrix (6.9), which is done to assure that each row has a magnitude equal to one. Using Taylor's expansion, which identifies that the magnitudes will never be greater or less than one, the resulting magnitudes adjustments are described as follows:

$$X_n = \frac{1}{2}\left(3 - X_o \cdot X_o\right) X_o \tag{6.20}$$

$$Y_n = \frac{1}{2}\left(3 - Y_o \cdot Y_o\right) Y_o \tag{6.21}$$

$$Z_n = \frac{1}{2}\left(3 - Z_o \cdot Z_o\right) Z_o \tag{6.22}$$

where X_n, Y_n, and Z_n are the normalized results, which are then updated into the rotation matrix R.

Before the rotation matrix R can be used, the error in roll, pitch and yaw angle need to be calculated, which is done by using the measurement captured by the accelerometer and magnetometer. We start by using the accelerometer to calculate the error in the pitch and roll

angle, and update the proportional and integrator terms n and p, which are used by equation (6.13) to correct the drifting error before the gyroscope measurements are used.

$$a_m = \frac{\sqrt{a_x^2 + a_y^2 + a_z^2}}{gravity} \tag{6.23}$$

where a_m is the magnitude of the accelerometer vector a from equation (6.2), which is scaled by gravity. The decision to use integral gain or proportional gain is determined by whether the gravitation force returned from the magnitude a_m is greater than 0.5 and less than 1.5. This is done using a_w as a multiplied weight of 1 or 0 against the integral or proportional gain.

$$e_{rp} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} r_{xz} \\ r_{yz} \\ r_{zz} \end{bmatrix} \tag{6.24}$$

After the error in the roll and pitch e_{rp} is calculated, this needs to be scaled onto the proportional gain constant p and added to integral gain n from equation (6.13).

$$p = e_{rp}.KP_{rp}.a_w \tag{6.25}$$

$$n = n + e_{rp} \times KI_{rp} \times a_w \tag{6.26}$$

where KP and KI are the fixed gains required for the PI controller. Now that error in the pitch and roll angles is accounted for, we need to deduce the error in heading angle.

$$e_h = r_{xx}.\sin(\psi) - r_{xy}.\cos(\psi) \tag{6.27}$$

$$e_{yaw} = \begin{bmatrix} r_{xz} \\ r_{yz} \\ r_{zz} \end{bmatrix}.e_h \tag{6.28}$$

where e_h from equation (6.27) describes the error in the heading angle, which is then used to apply the correction to XYZ rotation of the sensor, which is described by e_{yaw} in equation (6.28)

.

$$p = p + e_{yaw}.KP_{yaw} \tag{6.29}$$

$$n = n + e_{yaw}.KI_{yaw} \tag{6.30}$$

where KP_{yaw} and KI_{yaw} are the fixed gains required for the PI controller, and that concludes the calculation for errors in Euler angles, pitch, roll and yaw.

The Euler angles pitch, roll, and yaw, θ, ϕ, and ψ respectively, are evaluated as follows:

$$\theta = -\arcsin\left(r_{xz}\right) \tag{6.31}$$

$$\phi = \arctan\left(\frac{r_{zy}}{r_{zz}}\right) \tag{6.32}$$

$$\psi = \arctan\left(\frac{r_{xy}}{r_{xx}}\right) \tag{6.33}$$

The variables used to calculate the Euler angles are derived from rotation matrix R after the drift and error compensation have been accounted for.

6.4 An Analysis of the 9-axis IMU Sensor for Self-localization

To evaluate the effectiveness of the directional cosine matrix method, a hardware platform that can communicate with sensors and process data, and then communicate that data, is required. Another accurate method that returns the Euler angles is required to compare against the directional cosine matrix method.

The Arduino Pro Mini was selected because it has a high baud rate I2C bus for interfacing sensors and has a serial port for communication via Bluetooth. The microcontroller also needs to have the processing power to return a high enough sample rate, which the 16MHz Arduino does at about 15 samples per second, which is about 67Hz. The control software has been documented in the Appendix.

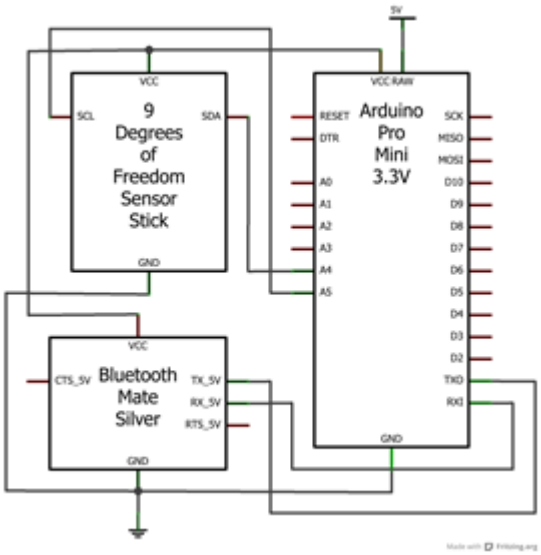


Figure 6-3 DCM Electronic Circuit Schematic

Using the specification needed to implement the directional cosine matrix method, the electronic circuit schematic in Figure 6-3 was drafted to trial the feasibility of the method.

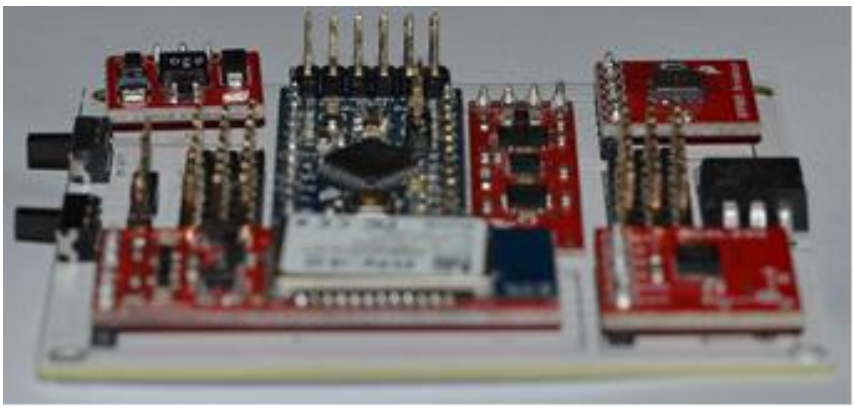


Figure 6-4 DCM Development Circuit Board

A development board was manufactured, as depicted in Figure 6-4, to carry out the experiment. The sensor board was then attached to a glyph icon.

Figure 6-5 DCM Glyph Tracking Setup

Tracking the glyph from a video camera, as depicted in Figure 6-5 using the AForge.NET Glyph Recognition and Tracking Framework (GRATF) toolkit [79], we can get the Euler angles to compare against the angles returned from the development sensor board.

Comparing the difference between the sensor board angles and the angles returned using the video based glyph tracking method, the following descriptive statistics were produced.

Method	**Error**
Mean	-0.0222
95% Confidence Interval for Mean (LB)	-0.1024
95% Confidence Interval for Mean (UB)	0.0579
Median	0.0308
Variance	1.1820
Standard Deviation	1.0872
Standard Error Mean	0.0483
Minimum	-3.0372
Maximum	2.9357
Range	5.9729
Interquartile Range	1.5628
Skewness	-0.0650

Table 6-1 Statistical analysis of DCM experimental data

The analysis of experiment is presented in Table 6-1, which is based on about 700 samples taken from a pitch movement experiment. The results show that the mean value is relatively low and, for more precision, because the data might be skewed, the median value is evaluated as it compensates for skewed data. The confidence intervals, which represent two standard deviations from the mean, are both relatively high, making it easy to say the rate of error is low.

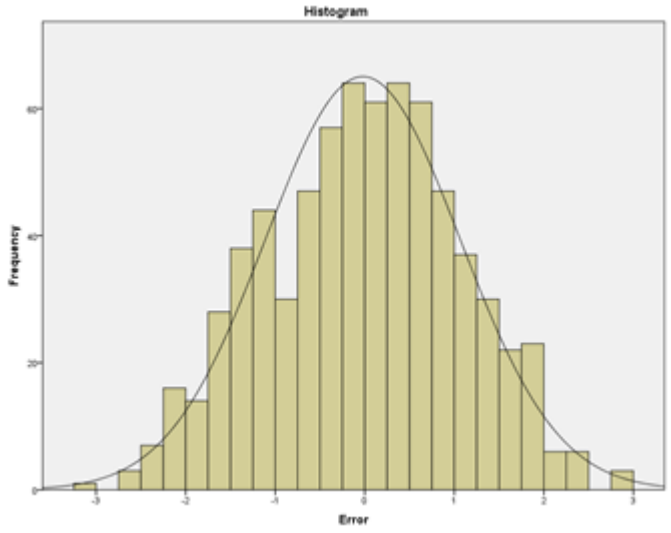


Figure 6-6 DCM Error Distribution Histogram

The error distribution is presented in the histogram in Figure 6-6, which shows how the error is Gaussian distributed, and with the peak errors focused around 0.

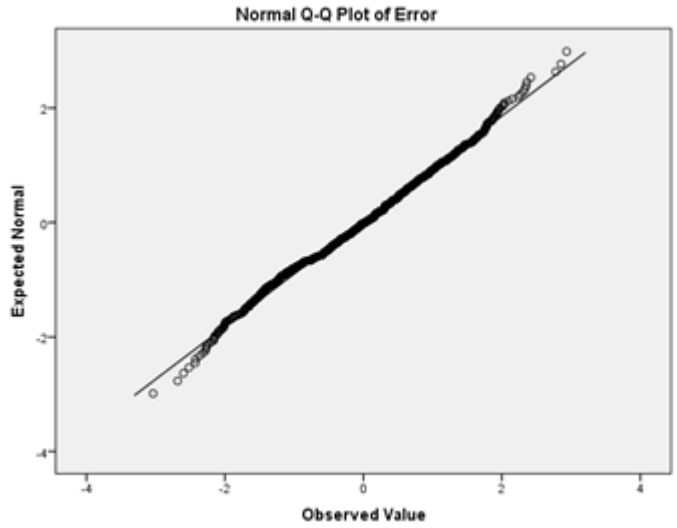


Figure 6-7 DCM Normal Q-Q Plot of Error

The normal Q-Q plot is depicted in Figure 6-7, which presents the performance of the observed values against the expected values. This shows that the errors are approximately normally distributed centrally, with anomalies at both tail ends.

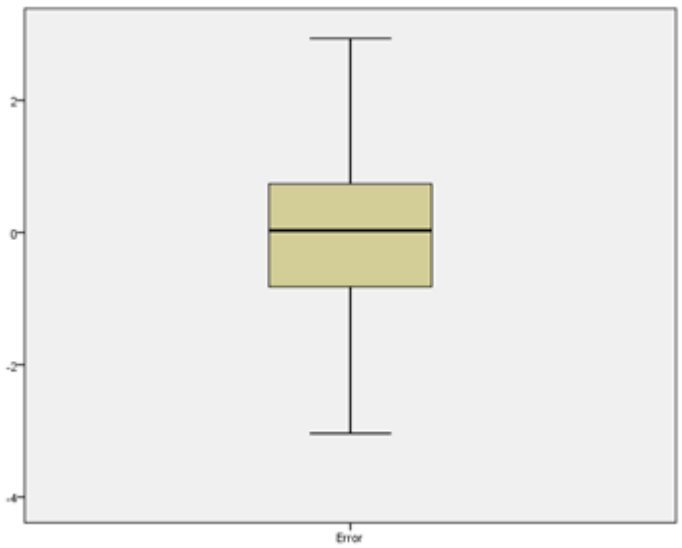


Figure 6-8 DCM Boxplot of Error Distribution

The boxplot in Figure 6-8 shows visually how close the mean is to 0. It also presents the interquartile range, which is 1.562864, with a minimum error value of -3.037223 and maximum error of 2.935766.

Using the statistical analysis of descriptive statistics presented in Table 6-1 and Figure 6-6 to Figure 6-8, we can show a good performance and high accuracy by employing the directional cosine matrix for self-localization. To solidify the findings, a non-parametric test is required to validate the practical effectiveness of the directional cosine matrix method. The ideal analysis test for a non-parametric independent one-sample set of data is the Kolmogorov-Smirnov test [80] for significance. For the experimental data presented in this paper, the null hypothesis is that the distribution of error is normal with a mean value of -0.022 and a standard deviation of 1.09. Based on the significance level of 0.05, a significance of 0.429 is returned using the one-sample Kolmogorov-Smirnov test. The strength of the returned significance value allows us to retain the null hypothesis and say that the distribution of error is normal with a mean value of -0.022 and a standard deviation of 1.09.

6.5 Summary & Conclusions

This chapter provided a novel optimized PI controller based on a directional cosine matrix to solve the self-localization of a mobile using 9-axis sensor, which is a fusion of an accelerometer, a magnetometer, and a gyroscope. The following research outcomes were presented in this chapter:

- The configuration for the 9-axis sensor was derived using an accelerometer, a magnetometer, and a gyroscope to form an inertial measurement unit.
- The model for the sensors frame of reference and the directional cosine matrix was derived.
- The solution to the systems pose was derived using the previously derived frame of reference and directional cosine matrix.
- A closed-loop optimized PI controller was developed to remove the erroneous drift caused by the sensor.
- A statistical analysis with a null hypothesis is validated to show the self-localization results from closed-loop PI controller with the directional cosine matrix method against a 3D computer vision tracking system.

The novel closed-loop PI control for the 9-axis fusion sensor delivered favourable results in the statistical analysis, showing a mean error of 1.56 degrees, which is very favourable. Based on these initial results this method has great potential. When compared to the Kalman filter method as presented by Ferdinando et al [81], which is heuristic and dependent previous results, which makes this method numerical and extremely computational. The complementary filter method as presented by Fourati [82], is a lightweight method using a high pass and low pass filter mechanism to selectively use the sensor that is trusted the most. While this method is not as computational as the Kalman filter method, it is significantly less accurate. The directional cosine matrix method is the ideal method for small mobile robots, as it is less computational than the Kalman filter method and significantly more accurate than the complementary filter method. It should also be noted that this method is not only limited to two-wheel manoeuvrable mobile robots and can be applied to any system that requires accurate self-orientation feedback. For example, this can be used for a self-balancing robot, a quad-copter, or even in a digital camera to stabilize the image sensor.

The statistical analysis presented favourable results, with a mean error that was less than ±1 degree. The research into the directional cosine matrix approach with the PI controller offers a valuable contribution to mobile robot research, offering an alternative to the computational Kalman filter method and the less accurate complementary filter approach.

For the 9-axis sensor to work, the individual sensors need to be calibrated; if the sensors remain un-calibrated, the closed-loop PI controller will return erroneous results. The accelerometer is sensitive to vibrations and an ideal g-force limit should be applied to filter out noise. The magnetometer is biased towards the earth's magnetic pole and should be calibrated to compensate for interference from hard and soft irons. The gyroscope needs to be levelled and the average value of this process needs to be applied to the measurements returned by the gyroscope sensor.

The next stage of this research is to develop a self-balancing controller using the closed-loop PI controller based on the directional cosine matrix. This will enhance the behaviour of the closed-loop PD controller presented in Chapter 4 and will help prevent the two-wheel manoeuvrable mobile robot from slipping. There is also potential interest in adapting this feedback into a quad-copter and evaluating how it performs compared to the Kalman filter method and the complementary filter method.

Chapter 7 A Pseudo Random Algorithm with a Gradient Policy for Overhead Camera Mobile Robot Tracking

7.1 Introduction

In Chapter 4, research on a closed-loop optimized PD controller was presented that used the two-wheel manoeuvrable mobile robot's pose and orientation as linear feedback. Followed by research in Chapter 5, on a novel self-localization method using a double compass configuration, further improved the linear feedback for closed-loop optimized PD controller from Chapter 4. In order to validate the behaviour of the closed-loop optimized PD controller and the novel self-localization method, a tracking system was required.

A highly accurate, easy to implement, with minimal computational and cost effective method was required to validate the functionality and performance of the closed-loop optimized PD controller and the novel self-localization method. To achieve this, it was ideal to use a computer vision method. This method was required to track the mobile robot's movement from an overhead camera system, while collecting the mobile robot's pose and orientation and synchronized time points. In order to identify the mobile robot's pose and orientation, non-natural coloured markers were used on the mobile robot. Using a computer vision method like this is known to return the most accurate results, but processing the images using a systematic search algorithm to find the mobile robot's pose and orientation is very time consuming and computationally expensive, and the analysis required thousands of images to be processed. A better method was required to improve the time it took to scan the images. This was achieved using a pseudo random search algorithm, based on a skip-list method, and gradient policy to improve the search and identification time of the non-natural markers in the images. To achieve an improved solution, the following objectives were set:

- Develop a platform to collect synchronized data from the image capture system and the mobile robot's self-localization feedback.
- Derive a pseudo random algorithm, inspired by the skip-list method, to search an image for non-natural markers and return the pose and orientation of the mobile robot.

- Derive a gradient policy in the pseudo random algorithm to further improve the search time by guessing the mobile robots next location in the image.

Using a systematic method that is computational and expensive is not an ideal solution and lead to the requirement to develop an improved method without sacrificing the accuracy of the scanning results.

The system is highly dependent on the use on non-natural markers, and the system requires calibration of these colours before the image scanning process can begin. The results are affected if the non-natural marker colours are not unique in the scanning environment. The resolution of the results is limited to the size of the markers which is coupled to the result's tolerance, which is 10mm.

7.2 Pseudo Random Decisions for Path Planning

The novelty of this section presents a skip-list inspired search algorithm method with a gradient policy to scan an image to localize the manoeuvrable mobile using two non-natural colour-marker tags. The skip-list was invented by William Pugh in 1990 and is described as a probabilistic data structure that seems likely to supplant balanced trees as the implementation method of choice for many applications, while the algorithms have the same asymptotic expected time bounds as balanced trees and are simpler, faster and use less space [83].

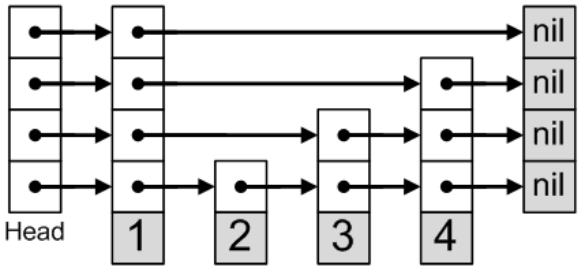


Figure 7-1 Skip-list example using a hierarchy of linked lists

Figure 7-1 is a schematic picture of the skip-list data structure. The boxes with arrows represent a pointer and each row is a linked list, which gives a sparse subsequence. The numbered boxes at the bottom represent the sequence the data is ordered in. The general idea is searching proceeds downwards from the highest subsequence at the top until consecutive elements linking the search elements are found.

A skip-list is built in vertical layers, as depicted in the Figure 7-1. The bottom layer is an ordinary ordered linked list. Each higher layer acts as an express lane for the lists below, where an element in layer i appears in layer $i+1$ with a fixed probability p. On an average basis, each element appears in $1/(1-p)$ lists, and the tallest element, described by the head element at the front of the skip-list, in $\log_{1/p} n$ lists.

The search for a target element begins at the head element in the top list, and proceeds horizontally, until the current element is greater than or equal to the target. If the current element is equal to the target, it is then found. If the current element is greater than the target,

the procedure is repeated after returning to the previous element and dropping down vertically to the next lower list. The expected number of steps in each linked list is at most $1/p$, which can be seen by tracing the search path backwards from the target until reaching an element that appears in the next higher list or reaching the beginning of the current list. Hence, the total expected cost of a search is $(\log_{1/p} n)/p$, which is $O(\log n)$ when p is a constant. By choosing a different value for p, it is possible to trade search costs against storage costs.

```
Search(list, searchKey)
    x := list→header
    -- loop invariant: x→key < searchKey
    for i := list→level downto 1 do
        while x→forward[i]→key < searchKey do
            x := x→forward[i]
    -- x→key < searchKey ≤ x→forward[1]→key
    x := x→forward[1]
    if x→key = searchKey then return x→value
        else return failure
```

Figure 7-2 Skip-list Search Algorithm Function Syntax

This search method is applied to the input image from the visual camera. The image is set up as a hierarchy of linked lists with the data being the pixel colour.

It is not ideal to have numerical solutions to the position and orientation of the mobile robot, as the computational requirement affects the solution being available in real-time and has an accumulative effect on the error and the solution result. Heavy computation also affects the battery life of the mobile robot, and this is a critical aspect. The ideal solution will not have accumulation of drift error and will not be dependent on the previous configuration values of position and orientation. Such a solution is only available by exploiting visual odometry methods.

7.3 Implementing Visual Odometry with an Overhead Camera

Using any standard camera at a resolution of 320 by 240 pixels, the system uses this device to capture images that are then used to compute the position and orientation of the manoeuvring mobile robot.

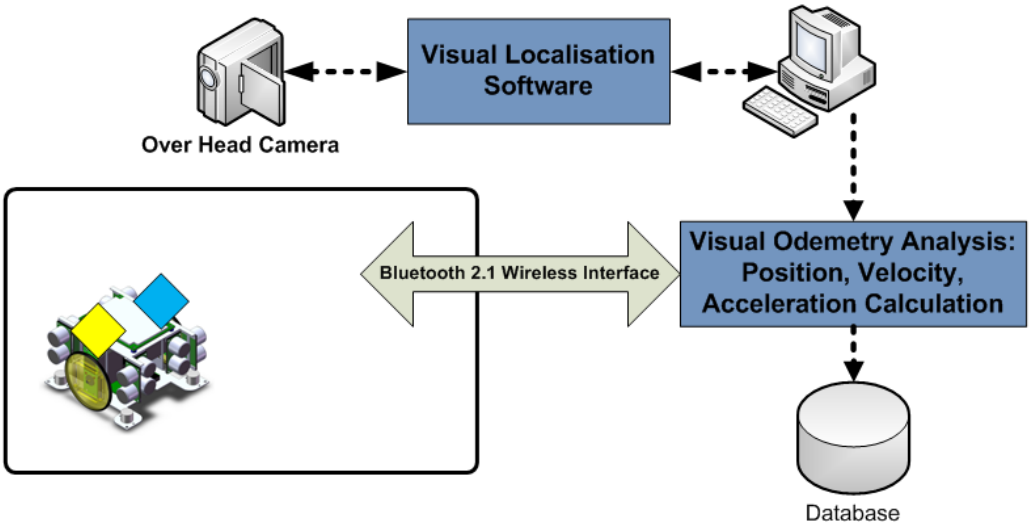


Figure 7-3 Visual Localization System Setup

The camera is positioned overhead of the manoeuvrable mobile robot at a fixed height. The camera is connected to an ordinary desktop computer that processes the images captured from the camera. The captured images are then processed to find the two markers on the mobile robot. The markers are positioned directly above the mobile robot's rolling wheels. The processing software then scans the current image, identifying the marker positions. Having two markers allows the software to deduce the orientation of the manoeuvring mobile robot. When the software has deduced the position and orientation of the mobile robot, this telemetry is communicated to the mobile robot via wireless Bluetooth signal. An overview of the setup is represented in Figure 7-3.

7.4 Systematic Searching vs. a Skip-list Inspired Searching

By using a systematic searching algorithm, the computation cost-time is dependent on the location of the markers. The closer the markers are to the origin (e.g. coordinates (0 pixels, 0 pixels)) of the search, the faster the search for the markers will be performed. The systematic search algorithm is represented as follows:

```
For x-pixel = 0 To Image width
   For y-pixel  = 0 To Image height
         If current pixel (x-pixel, y-pixel) = Marker 1 Then
                  Marker 1 x position = x-pixel
                  Marker 1 y position = y-pixel
                  Flag = Flag + 1
         End if
         If current pixel (x-pixel, y-pixel) = Marker 2 Then
                  Marker 2 x position = x-pixel
                  Marker 2 y position = y-pixel
                  Flag = Flag + 1
         End if
         If Flag > 1 Then
                  x-pixel = Image width
                  y-pixel = Image height
         End if
   Next
Next
```

The search algorithm above shows how every vertical and horizontal pixel is scanned to identify the two defined markers on the mobile robot. When a marker is found, a flag is raised. When the search has two raised flags, the search will end.

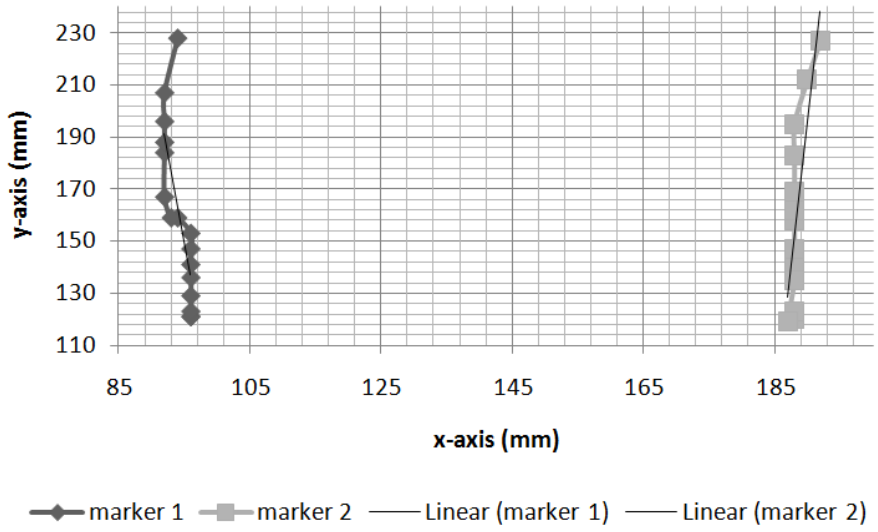


Figure 7-4 Implementation of Systematic Search Algorithm

Figure 7-4, demonstrates a sample of the implemented systematic search algorithm on the mobile robot, making a forward manoeuvre over a length of about 100mm.

The fact that the systematic search needs to scan every pixel makes the scanning process non-ideal if a high resolution image is required, and also affects the resolution of the telemetry sent to the manoeuvring mobile robot. As such, a more effective process to reduce the number of pixels scanned is introduced using William Pugh's skip-list search method to inspire a new method for finding pixel markers. The algorithm is presented as follows:

```
For y-axis = 1 To log2(image height)
    y_pixel = 0
    For y = 1 To 2^(y−axis)
        y_pixel = y_pixel + image height / 2^(y−axis)
            For x-axis = 1 To log2(image width)
              x_pixel = 0
              For x = 1 To 2^(x−axis)
                x_pixel = x_pixel + image width / 2^(x−axis)
        If current pixel (x-pixel, y-pixel) = Marker 1 Then
                Marker 1 x position = x-pixel
                Marker 1 y position = y-pixel
                Flag = Flag + 1
        End if
        If current pixel (x-pixel, y-pixel) = Marker 2 Then
                Marker 2 x position = x-pixel
                Marker 2 y position = y-pixel
                Flag = Flag + 1
        End if
        If Flag > 1 Then
                y = 2^(y−axis)
                x = 2^(x−axis)
                y-axis = log2(image height)
                x-axis = log2(image width)
        End if
        Next
        Next
   Next
Next
```

The search algorithm above shows how the inspired skip-list algorithm is implemented to identify the two defined markers on the mobile robot. When a marker is found, a flag is raised. When the search has two raised flags, the search function will end.

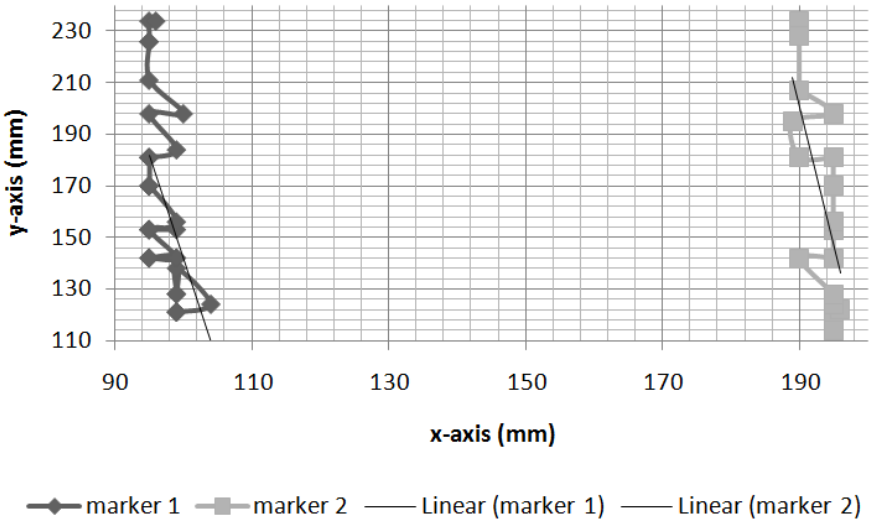


Figure 7-5 Implementation of Skip-list Inspired Search Algorithm

Figure 7-5 demonstrates a sample of the implemented inspired skip-list search algorithm on the mobile robot, making a forward manoeuvre over a length of about 100mm.

When comparing the tracking results of the systematic search, Figure 7-4, against the inspired skip-list search, Figure 7-5, it should be noted that variable accuracy of the marker position is affected by the new algorithm. The typical resolution of accuracy is with 10mm, this is based on the markers width and height, and it can be seen that in both tracking results that this holds true, with the systematic search producing a smoother plot of position.

It is also fundamentally important to consider the computation's cost of implementing a different search algorithm. Ideally, the computation for the mobile robot's location will be done locally on the mobile robot platform where the computational power is limited by clock speed of the processor. The motivation for an alternative search method is a low computational cost method, to facilitate the local mobile robots processor limitations.

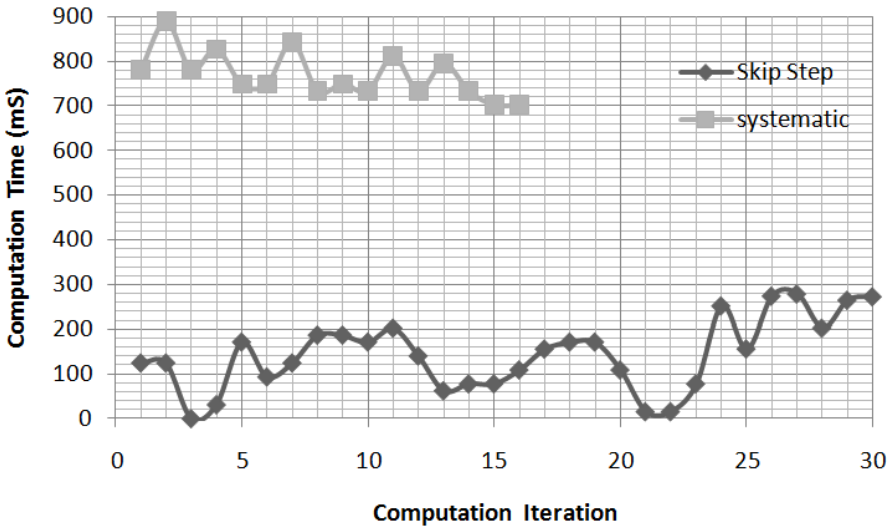


Figure 7-6 Algorithm Computation Cost Comparison

The graph above, Figure 7-6, depicts the clear improvement on the computation cost of using the inspired skip-list algorithm. The search time for a systemic search is 769.2mS and for the inspired skip-list method, it is 153.6mS, which represents an improvement of 615.6mS. This verifies the benefit of introducing this search algorithm over the systematic based algorithm.

To improve the search area, the velocity of both markers are monitored and a gradient policy is introduced to reduce the area searched.

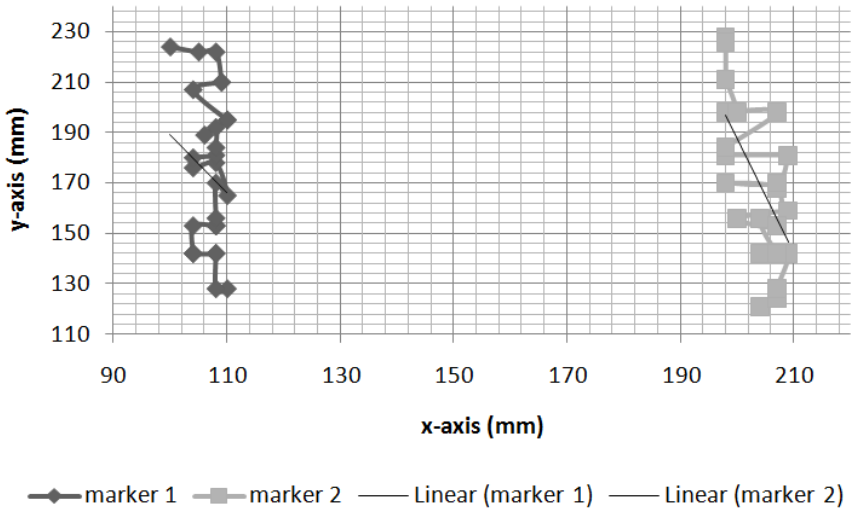


Figure 7-7 Implementation of Skip-list Inspired Search Algorithm with a Gradient Policy

Figure 7-7 demonstrates a sample of the implemented inspired skip-list search algorithm with a gradient policy on the mobile robot making a forward manoeuvre over a length of about 100mm.

Comparing the application of the gradient policy tracking, as depicted in Figure 7-7, to the tracking results of without the gradient policy, Figure 7-5, it is clear that resultant plot is slightly more noisy but still with the tolerance of 10mm. Even though the results are not as accurate as the systematic plot results, the values are still correct within the defined resolution of the marker size, which is 10mm.

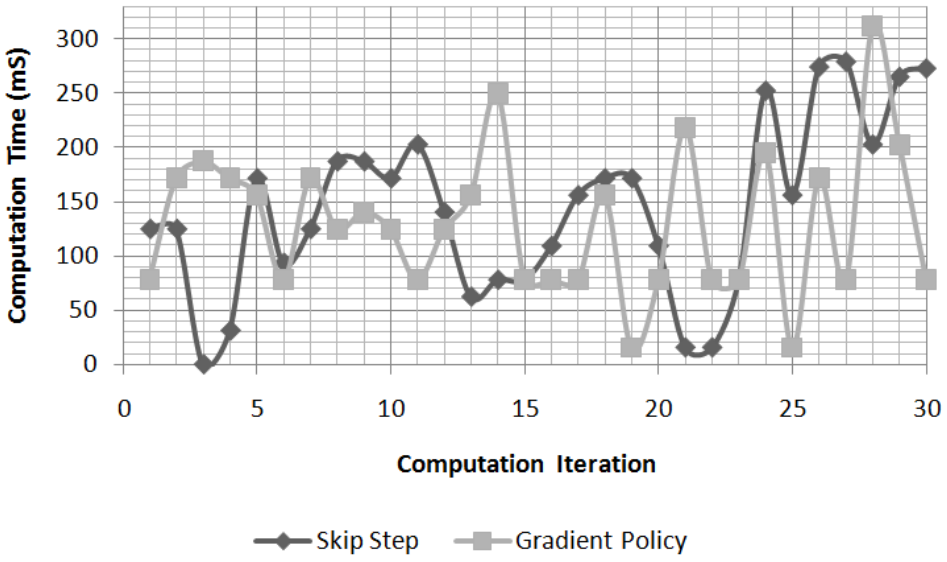


Figure 7-8 Algorithm Computation Cost Comparison with Gradient Policy

From the computation cost comparison, Figure 7-8, it is not clear if any performance benefit is offered by introducing a gradient policy. As previously stated, the average computation cost of inspired skip-list algorithm is 143.6mS, where by introducing the gradient policy, the results show a 130.78mS average computational cost. The results are clear that the gradient policy has further reduced computation time by 12.8mS.

7.5 Summary & Conclusions

In this chapter, research on a novel pseudo random algorithm with a gradient policy was derived to assist the image processing task required for the analysis to validate the behaviour of the closed-loop optimized PD controller from Chapter 4 and the novel self-localization method from Chapter 5. The following research outcomes were presented in this chapter:

- An overhead camera system was developed for tracking mobile robots with non-natural markers.
- A pseudo random, skip-list inspired, algorithm with a gradient policy was derived.
- The performance of the algorithm was validated.

The research in this chapter produced an algorithm that can be used to scan images for non-natural markers and return a pose and orientation for mobile robot tracking. This algorithm can be used by other researchers with the requirement to validate a problem that requires an accurate estimate of the objects pose and orientation in a two-dimensional space.

The benefits of the novel pseudo random algorithm showed a 615.6mS improvement over the systematic search method. A further 12.8mS improvement was added by introducing a gradient policy to the search algorithm.

The system is highly dependent on the environment being free from the colour of the non-natural markers. The pose and orientation results are also limited by the size of the markers.

To resolve the dependency on the non-natural markers and improve the tolerance on the results, the system can employ a more complicated computer vision method that is developed to identify the object rather than the markers. This can be achieved using a training set of images and a method like principle component analysis (PCA) can be exploited to scan and find the mobile robot. While this method might return better results, it might also be more computationally expensive.

Chapter 8 Environment Map Building using a RGB-D Camera

8.1 Introduction

In Chapter 4, the two-wheel manoeuvrable mobile robot constraints and system model were derived with an optimised closed-loop PD controller with linear feedback. The controller gave the mobile robot the ability to follow a predefined path as quickly as mechanically and physically possible. In Chapter 5, the limitation of the linear feedback was addressed and a novel self-localization was derived to further optimize the behaviour of the closed-loop PD controller. Chapter 6 looked at the limitation of both the closed-loop PD controller and new novel self-localization method as not being able to compensate for slip. As a result, research was carried out to derive a new novel 9-axis sensor that would assist in configuring the KCLBOT as a self-balancing robot to prevent slip. Using the closed-loop optimized PD controller from Chapter 4, the novel self-localization method from Chapter 5 and the 9-axis inertial measurement unit for preventing slip from Chapter 6, a robust system for navigation is now possible. The next challenge that the mobile robot has to overcome, now that navigation is possible, is obstacle avoidance. Before the system is capable of obstacle avoidance, it will require information about the environment it is navigating in. With information about the environment, the mobile robot is tasked with getting from point A to point B as quickly as possible, which will require a path planning strategy.

Before the mobile robot can start the navigation task of going from point A to point B, the mobile robot needs to know about the environment it is traversing in. This is essential for two reasons- firstly, because without information about the environment, the system cannot optimize the path to arrive at point B as quickly as possible, and secondly, without information about its environment, the mobile robot has the potential to collide with obstacles and damage the obstacle and itself. Therefore, having information about the environment the mobile robot is going to traverse in, is a critical component of mobile robot navigation. In this chapter, research into environment map building using a RGB-D camera system is explored. To achieve environment map building using a RGB-D camera system, the following objectives are required:

- Derive, using machine learning methods, a model for converting RGB-D sensor data into a measurable unit of distance.

- Derive a model to translate the estimated distance data into a two dimensional environment map.
- Validate the distance estimation model using a high resolution range sensor against a static object at variable distances.

The environment map plays a critical role in mobile robot navigation, and without information about the environment, the mobile robot has the potential to collide with obstacles. Using the RGB-D sensor with a model to translate the RGB-D data into a navigation map, the mobile robot can avoid obstacles and select optimized routes to get to its destination as quickly as possible.

To enable the KCLBOT to be capable of carrying the RGB-D camera, the system needed an additional processor, as the Arduino platform was not capable of processing the data from the RGB-D camera. The VIA Pico-ITX system was added to the platform with the addition Gorilla Mini Power pack to meet the need of the new power consumption. It is important to note that the RGB-D camera is dependent on the environment being well lit; the camera system is not capable of returning data in the dark. The RGB-D camera is also sensitive to transparent materials like glass and will return erroneous results when scanning environments with these types of surfaces.

8.2 Machine Learning Distance Estimations Using a RGB-D Camera Data

The motivation for using a linear regression model for estimating distance based on a RGB-D sensor input is that it allows for the input of multiple features associated with the image capture. Some examples of the features that can be used are the exposure of a single shutter cycle, aperture size, lens focal length and ISO compensation.

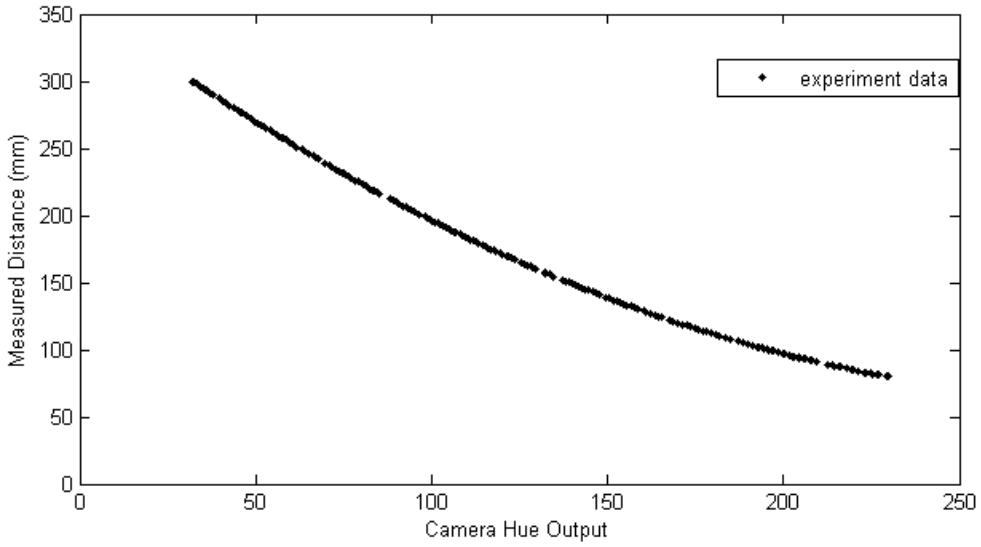


Figure 8-1 Fixed Object Capture – Experiment Data

Using the Microsoft Kinect RGB-D sensor [34], 220 image samples were taken of a fixed object between 800mm and 3,000mm. The fixed object was measured empirically assuming a tolerance of ±10mm. The results of the experiment are presented in Figure 8-1 above.

The objective of utilizing linear regression is to achieve the minimum valuation of the cos function via a numerical gradient descent method. The cost function $J(\theta)$ is derived on the sum of squares for error (SSE) methodology, where the cost function $J(\theta)$ is described as follows:

$$J(\theta) = \frac{1}{2m}\sum_{t=1}^{m}(h_\theta(\text{hue}^{(t)}) - d^{(t)})^2 \tag{8.1}$$

Where m describes the number of experiment samples, $h_\theta(x)$ describes the heuristic hypothesis of the behaviour of the camera hue values mapped as hue and d is the empirically measured distance values that correspond to the hue values mapped in hue. The hypothesis $h_\theta(x)$ is constructed using a linear model and is represented as follows:

$$h_\theta(\text{hue}) = \theta^T hue = \theta_0 hue_0 + \theta_1 hue_1 \tag{8.2}$$

Where θ describes the parameters of the model, which will be manipulated to minimize the cost function $J(\theta)$.

A common method for solving θ from equation (8.2) is by utilizing a batch gradient descent algorithm, which is represented as follows:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{t=1}^{m} (h_\theta(\text{hue}^{(t)}) - d^{(t)}) \, \text{hue}_j^{(t)} \tag{8.3}$$

Where θ_j is updated simultaneously for j values using an iterative approach to get θ_j to converge closer to the optimal value that will return the lowest cost from $J(\theta)$. The α scalar value is used as a learning rate to help converge θ_j to the lowest cost value. Typically, a small value is used to help the convergence process as a large value can affect getting to the local or global optimum value.

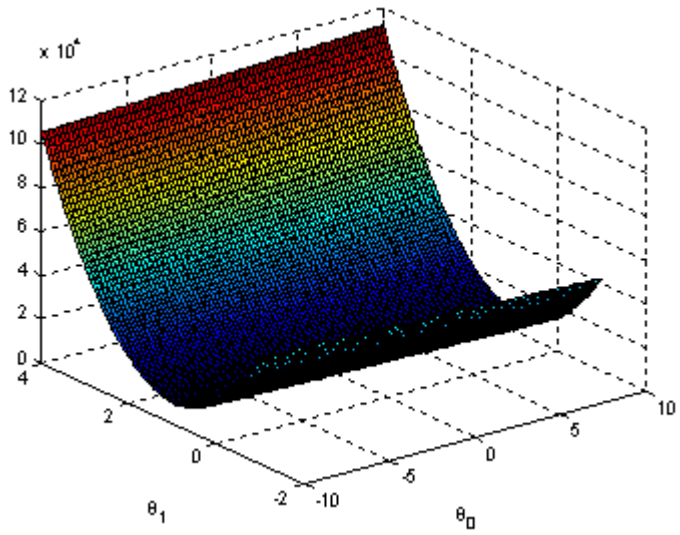


Figure 8-2 Gradient Descent Experiment Plot

With only two parameters, as in Figure 8-1, it is easy to visualize the cost function utilized by the gradient descent approach. When dealing with a higher order of parameters, it is more difficult to visualize gradient descent.

Having a cost function (8.1) and a gradient descent model (8.3), it is possible to complete the hypothesis (8.2).

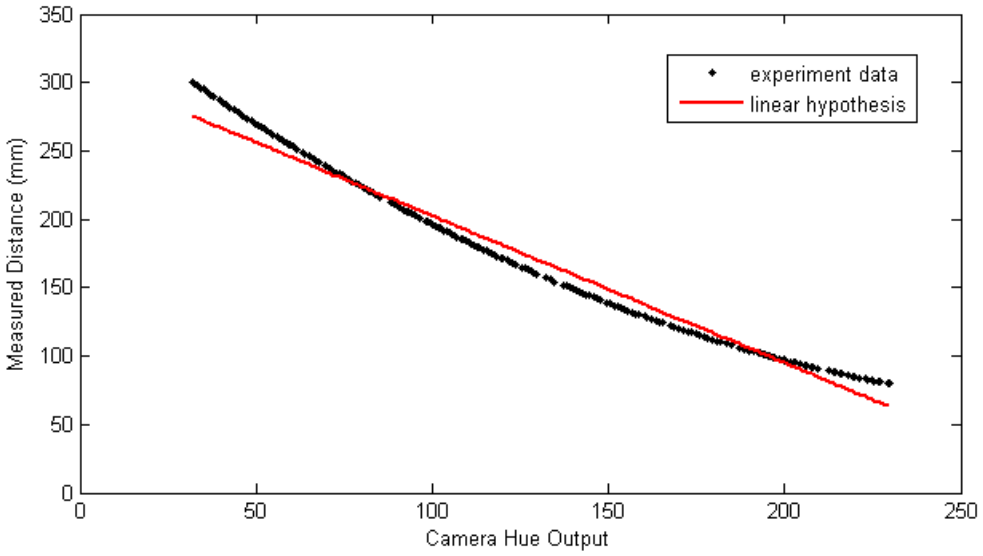


Figure 8-3 Linear Hypothesis for Experiment Data

From Figure 8-2, which is a plot of the hypothesis (8.2) compared to the experiment data, it can be observed that the hypothesis is under-fitting our experimental results. It would not be ideal to use this linear hypothesis to fit our experimental data and a polynomial based hypothesis should offer a more ideal fit to the available data.

$$h_\theta(\text{hue}) = \theta_0 + \theta_1 hue_i + \theta_2 hue_i^2 + ... + \theta_m hue_i^m \tag{8.4}$$

The hypothesis (8.4) presents a polynomial based equation, which will offer a more ideal fit to the experiment data. For this experiment, a second order polynomial is used to define the hypothesis. As the gradient descent approach is a very computation heavy method, to solve the θ parameters for the hypothesis it is more ideal to use a normal equation, which is the closed-form solution to the linear regression problem faced.

$$\theta = (X^T X)^{-1} X^T \vec{d} \tag{8.5}$$

Using the normal equation (8.5), the solutions to the parameters are computed and used to describe the polynomial based hypothesis.

$$h_\theta(\text{hue}) = 359.26 - 1.9492.\text{hue} + 0.0032.\text{hue}^2 \tag{8.6}$$

The polynomial based hypothesis (8.6) provides an accurate fit to the experiment data.

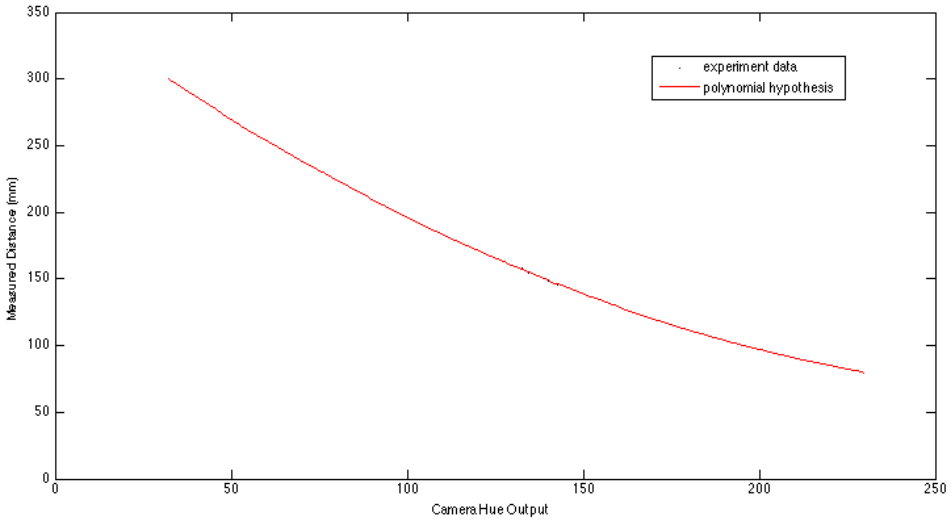


Figure 8-4 Polynomial Hypothesis for Experiment Data

To validate hypothesis (8.6), it is implemented in the cost function (8.1). This returns a negligible SSE. An alternative visual comparison is presented in Figure 8-3 Linear Hypothesis for Experiment Data.

In equation (8.2), only a single feature, hue, is considered, which is easily solved by an analytical solution (8.5), as presented. The image data from the sensor returns multiple features and, for this experiment, we will consider the colour space values red, green and blue, the image hue, saturation and the V value. Using more features allows for identify distance allows the prediction to have a higher accuracy. The hypothesis (8.2) will now consider the seven features of x_{0-6} and the linear regression gradient descent solver (8.3) for θ_{0-6}. For the

gradient descent solver (8.3) to work, the numeric values of the features having different ranges need to be normalized as follows:

$$x_i = \frac{x_i - \mu_i}{s_i} \tag{8.7}$$

where μ describes the average value of the feature and s is the standard deviation of the feature. If the features are not normalized (8.7), features with high numeric values will hold a high weight when the solver is implemented.

To assist the gradient descent (8.3) solver with many features, the cost function (8.1) also needs to be manipulated.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x) - d)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right] \tag{8.8}$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(t)}) - d^{(t)})x_j^{(t)} - \frac{\lambda}{m}\theta_j\right] \tag{8.9}$$

Where λ is a constant used to control the compensation strength of the regularization introduced in equations (8.8) and (8.9). It should be noted that the new gradient descent model is not implemented on the initial θ_0 variable. The main reason for introducing regularization is to prevent a hypothesis model that over-fits the experiment data.

By implementing the gradient descent solver, the following results are returned for θ:

θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
307.11	0.05	-0.127	0.0428	-1.171	-4.8	39.9296

Table 8-1 Gradient Descent Solutions

With the θ values in Table 8-1, the hypothesis is able to estimate distance based on the multiple features of the sensor image.

8.3 2D to 3D Environment Map Building Using a RGB-D Camera

Before the autonomous mobile robot can complete any path planning or path following tasks, it requires sufficient information about the environment that it will be traversing. A camera system alone with RGB output alone does not provide enough information about the environment and only returns a 2D map of pixels with colours. When this optically matched image in integrated with depth information of the environment, this combination is referred to as RGB-D, where the D refers to depth. To provide the mobile robot with this extra information, the detail from the RGB-D camera is used to make a plot of the terrain, mapping the unobstructed space the mobile robot can utilize.

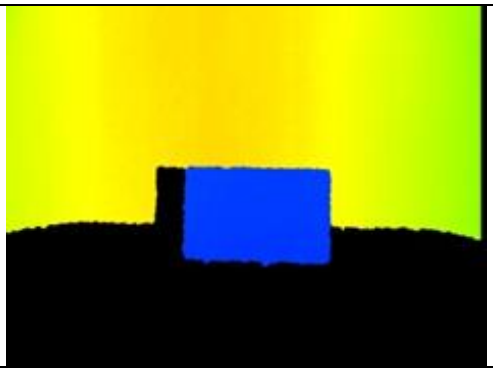
Before the RGB-D image can be used, the noise, resolved as black pixels in the range of #E4E1C0h to #FFFFFFh, needs to be removed from the image. This is achieved by converting the RGB-D image to grey scale [84]. This process is carried out to protect natural colours in the #E4E1C0h to #FFFFFFh range. In the RGB colour model, a colour image can be represented by the following intensity function:

$$I_{RGB} = (F_R, F_G, F_B) \tag{8.10}$$

In equation (8.10), F_R is the intensity of the pixel (x,y) in the red channel, F_G is the intensity of pixel (x,y) in the green channel, and F_B is the intensity of pixel (x,y) in the blue channel. Using only the brightness information, the colour image can be transformed into a grey scale image [84].

$$I_{GS} = 0.333F_R + 0.5F_G + 0.1666F_B \tag{8.11}$$

where equation (8.11) derives the conversion of a colour pixel to a grey scale pixel.



RGB-D Image

Grey Scale Image

Figure 8-5 Conversion from RGB-D to Grey Scale

After the image has been converted to grey scale, as depicted in Figure 8-5, the black pixels are filtered out of the image.

Figure 8-6 Filtering RGB-D Grey Scale Image

Once the image has been stripped of the black noise pixels, as depicted in Figure 8-6, the colour detail is required for mapping the traversable environment.

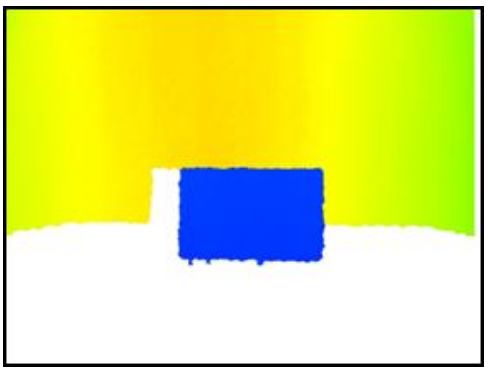


Figure 8-7 Remapping Colour to Filtered Grey Scale Image

The RGB-D depth colour information from Figure 8-5 is remapped onto the grey scale filtered image; the result is presented in Figure 8-7.

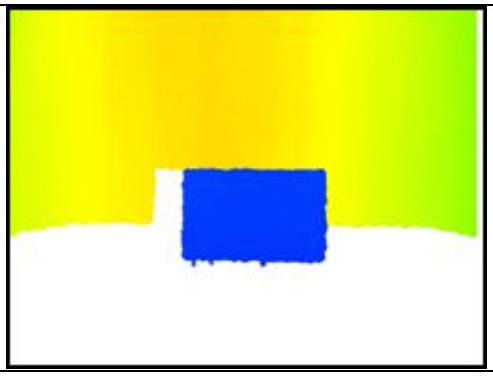
$$Input_{pixel} = \begin{bmatrix} x_{pixel} & y_{pixel} & RGB\text{–}D_{pixel} \end{bmatrix} \tag{8.12}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{8.13}$$

The input matrix (8.12) is multiplied by the rotation matrix [85] from (8.13), where $\theta = 90^{\circ}$, to produce a topological view of the traversable space.

$$Output_{pixel} = \begin{bmatrix} x_{pixel} & RGB\text{–}D_{pixel} & -y_{pixel} \end{bmatrix} \tag{8.14}$$

Using the first two components of (8.14), a 2-dimensional traversable space plot is produced, where the $RGB\text{–}D_{pixel}$ is calculated using (8.6).



Input Image

Output Image

Figure 8-8 Environment Map Generation from RGB-D Image

The generation of a topological view of the available navigation space, as presented in Figure 8-8, provides the mobile robot with sufficient information for path planning and obstacle avoidance

8.4 An Analysis of the RGB-D Sensor Application

Having a hypothesis from the regularized gradient descent solver, in Table 8-1, to validate the hypothesis, the error between the empirically measured values are compared to the output of the derived hypothesis.

Using an experiment sample size of 220 images with multiple features from the RGB-D sensor, the error difference is considered for the analysis.

Method	**Evaluation**
Mean	-0.0048
95% Confidence Interval for Mean (LB)	-0.1063
95% Confidence Interval for Mean (UB)	0.0966
5% Trimmed Mean	-0.0313
Median	-0.1523
Variance	0.5830
Standard Deviation	0.7636
Standard Error Mean	0.0514
Minimum	-1.3223
Maximum	2.1117
Range	3.4340
Interquartile Range	1.0723
Skewness	0.5340
Kurtosis	-0.4110

Table 8-2 Statistical Analysis of RGB-D Experiment Error Data

The descriptive statistics from Table 8-2 give a general idea of the performance of the hypothesis. The mean error value is negative 0.005, which is relatively low. From the presented descriptive statistics, it is difficult to tell if the data is skewed and as such, it is better to consider the median value which is negative 0.152, which is also comparably low. The confidence intervals, which are two standard deviations from the mean, show a low error rate.

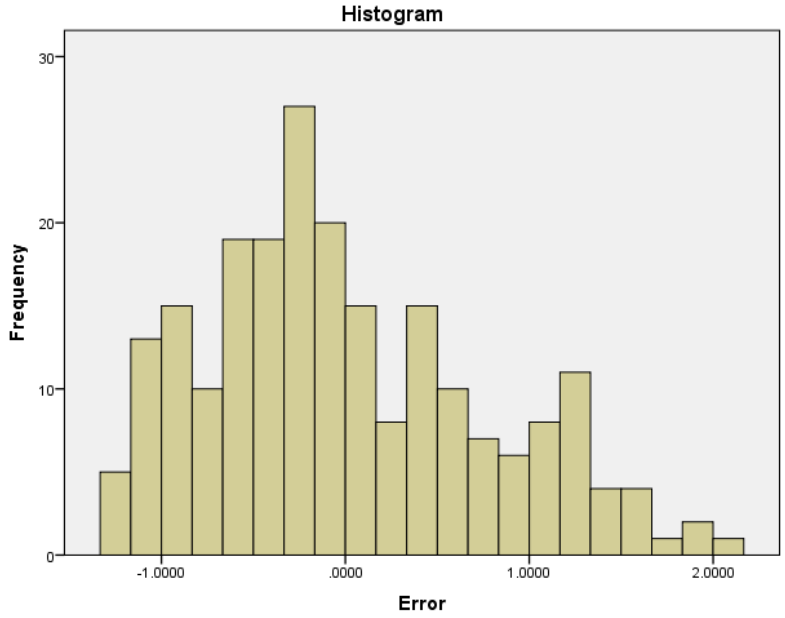


Figure 8-9 RGB-D Histogram: Frequency vs Error

The histogram in Figure 8-9 shows that error is roughly Gaussian distributed between -1.5mm and 2mm, with the peak very close to 0mm.

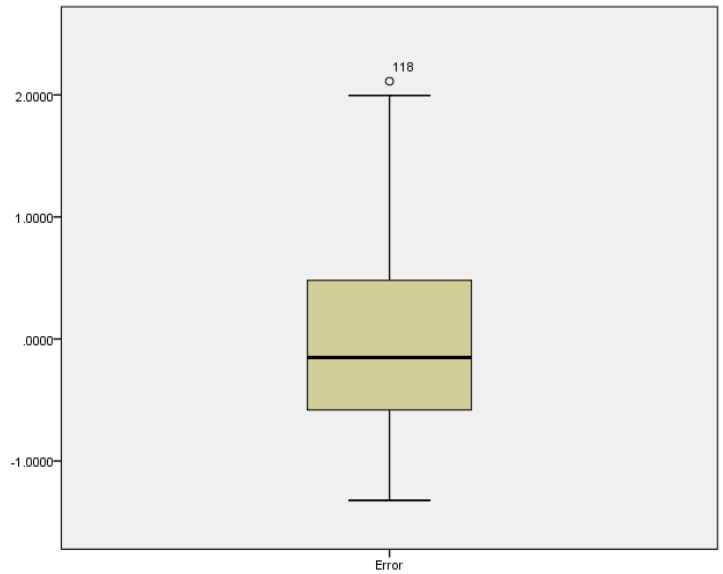


Figure 8-10 RGB-D Boxplot of the Hypothesis Error

The boxplot in Figure 8-10 highlights very favourable results, presenting the mean value close to 0mm and a very narrow interquartile range. The interquartile range is 1.0723mm and has a minimum value of -0.106mm and a maximum value of 0.097mm. An outlier is also present at data sample 118, which is ignored.

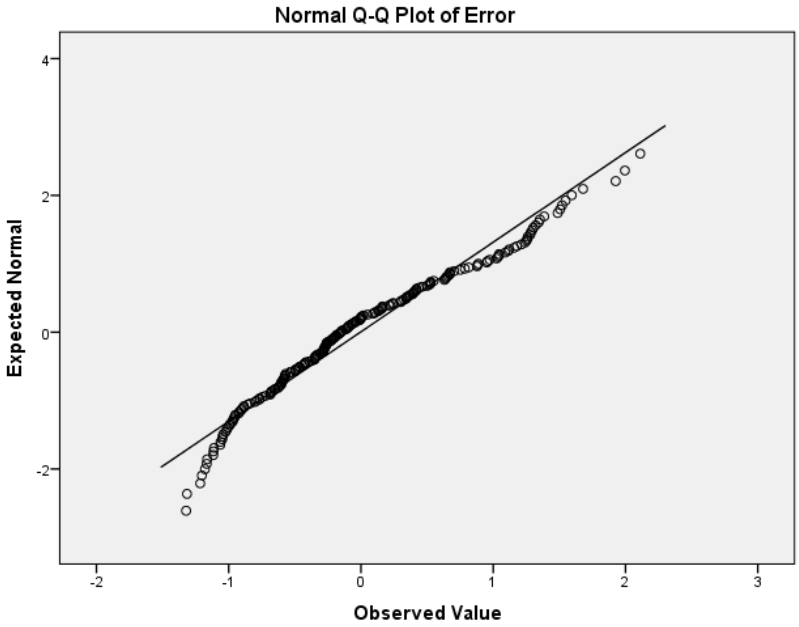


Figure 8-11 RGB-D Normal Q-Q Plot of Error

The normal Q-Q plot shown in Figure 8-11 shows the performance of the observed values against the expected values. The observation is that errors are normally distributed centrally, with anomalies at the tail ends.

Utilizing the descriptive statistics in Table 8-2, a non-parametric test is required to validate the effectiveness of the hypothesis, from Table 8-1, for the distance estimation model using multiple features. The ideal analysis test for a non-parametric independent one-sample set of data is the Kolmogorov-Smirnov test [67] for significance.

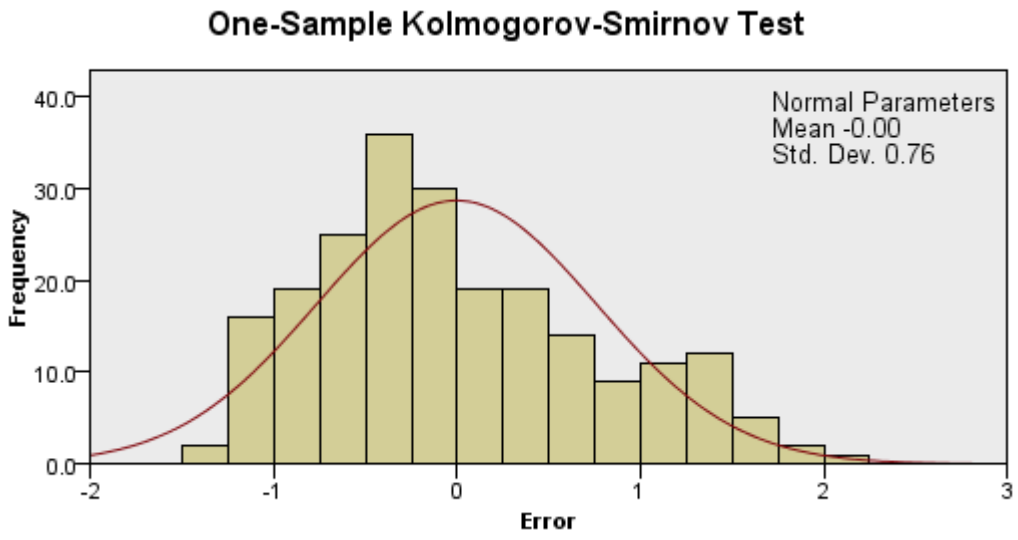


Figure 8-12 RGB-D One Sample Kolmogorov-Smirnov Test

For the experimental data presented in this paper, the null hypothesis is that the distribution of error is normal with a mean value of -0.005 and a standard deviation of 0.76. Based on the significance level of 0.05, a significance of 0.068 is returned using the one sample Kolmogorov-Smirnov test. The strength of the returned significance value allows us to retain the null hypothesis and say that the distribution of error is normal with a mean value of -0.0048 and a standard deviation of 0.7636.

8.5 Summary & Conclusions

In Chapter 8, a model is presented using a regularised linear regression model with a gradient descent solver to resolve the optimum behaviour of an RGB-D sensors output. The optimum polynomial model is used to build a 2-D environment map using a novel algorithm based on a rotation matrix. A full statistical analysis is carried out to infer that the polynomial model is valid. The following research outcomes were presented in this chapter:

- A novel model to transform RGB-D sensor data into a measurable distance estimate is derived.
- A conclusive statistical analysis is presented to validate the behaviour of the transformation equation.
- A rotation matrix based algorithm is derived to transform the RGB-D sensor data into a two-dimension environment map.
- Experimental data is presented to validate the performance of the rotation matrix based algorithm.

Navigation environment maps are essential to the mobile robot because without them, there is the potential for obstacle collisions that will damage both the mobile robot and the obstacle, and will most likely prevent the mobile robot from reaching its target goal position. Navigation environment maps also give the mobile robot the ability to optimize its path planning. The machine learning method to derive an estimate of distance from the RGB-D sensor is a contribution to researchers trying to exploit this sensor for environment map building. This research is valuable to the mobile robot community wishing to add on-line environment mapping to their mobile robot platforms.

The research in this chapter contributes significantly to the mobile robot community by providing a validated model for transforming RGB-D sensor data into a measurable distance estimate. This can then be used with the algorithm to transform this sensor data into a two-dimensional environment map. Mobile robots need to be aware of their environment to avoid obstacles and to optimize their path to their target goal position. This research is especially valuable to mobile robot platforms that do not have the ability to sense their environment or for those systems that use off-line environment maps. It is also noteworthy that this research is applicable to small

mobile robot platforms, but not limited to them, and can be adapted to any mobile robot platform.

While the performance of the RGB-D sensor is a contribution to mobile robot environment map building research, as inferred by the statistical analysis, this does not come without a cost. The camera system is quite large, maybe as large as the mobile robot itself, as depicted in Figure 2-6. The weight of the camera system can affect the mobile robot's centre of gravity and, as a consequence, this will add to the performance issues of the PD controller model discussed in Chapter 4. The camera system requires considerable numerical computation power to perform the computer vision task, and this was not possible using the Arduino microcontroller alone. As such, the more powerful VIA Pico-ITX computer system was added to the mobile robot platform to make this research possible. It should be noted that the added computer system and camera vision system will also draw considerable power, requiring a larger battery source as a result.

The statistical analysis presented some good results, showing that the sensor performed within an accuracy of ±2mm, but sensor results were limited to perform only between 800mm and 3000mm. PrimeSense have rereleased a new sensor that is capable of measuring at a minimum distance of 400mm. This, however, remains an issue for small mobile robots, as they are typically expected to perform tasks in a small environment and this minimum value will prevent this sensor from being a contribution. While this is an issue, there are solutions; it is known that with the addition of a focal adjusting lens, the sensor can be manipulated to perform at a lower distance. This will lead to further research, outside of the scope of this thesis, that will be valuable to the community.

Chapter 9 Stereo Vision for a Small Mobile Robot

9.1 Introduction

In the previous chapter, Chapter 8, the research covered a novel approach to environment map building using an RGB-D camera system with a machine learning distance estimation method that is applied to rotation matrix based algorithms to build two-dimensional environment maps. While this research produced favourable results with a distance accuracy of ±1mm (two standard deviation interquartile ranges), the technology for the RGB-D camera system was too large for the KCLBOT mobile robot. The research in this looks at alternative methods for environment mapping that have smaller form factors, which will consume less computation power, and will draw less power from the battery power cells.

There are many different research approaches that can be followed to achieve environment mapping. Like the research by Park et al [86] that proposes the use of multiple IR sensors to build environment maps. Nevertheless, this research has its limitations and requires that the mobile robot does at least one 360-degree rotation before it has a map; this is not efficient enough for an effective real-time mapping solution. The research presented by Tomono [87], as well as the research by Xu et al [88], shows how a 2D laser range finder can be used to scan an area and build an environment map. Again, this type of system requires a waiting time for the laser to scan an area and then process the data to the system, which is not ideal for real-time feedback to detect moving objects in unknown environments. It is also important to consider the cost of research; laser range finders alone cost more than the mobile robot, and this make this un-accessible to a majority of researchers. Finally, as far as the laser range finder is concerned, power consumption is important and the laser system draws too much current, requiring a significant power source to make this a feasible research option. In terms of limitations of the range sensors, like an IR or an ultrasound range sensor, which are good for detecting obstacles at discrete positions, when applied to build environment maps, they require too much lag time to be effective. The alternative research option is to consider stereo vision using off-the-shelf cameras. USB cameras are affordable, they do not take up a significant amount of space, do not consume an abundant amount of power, and they do not have a significant weight. All these factors - light weight, power efficiency and economic affordability - make this an ideal research

topic for adapting to mobile robot environment mapping. To prepare the mobile robot platform for environment map building using a stereo camera system, the following objectives are required:

- Develop a stereo vision capable mobile robot platform.
- Derive the camera geometry that models the orientation and position of the image point to the object point.
- Derive the camera calibration methods to remove radial distortion and skewness from the camera images.
- Solve the projection's matrix using the homography matrix with the intrinsic and extrinsic parameters on the stereo image pairs.

While the camera system is more affordable, has a smaller form factor, is lighter, and uses less power than the RGB-D camera system in Chapter 8, the system still requires significantly more computational power than the Arduino can provide.

9.2 Mobile Robot Configuration

An autonomous mobile robot platform that desires to carry a stereo vision system requires certain hardware, which usually includes two or more cameras and a high level processor to manage the image acquisition.

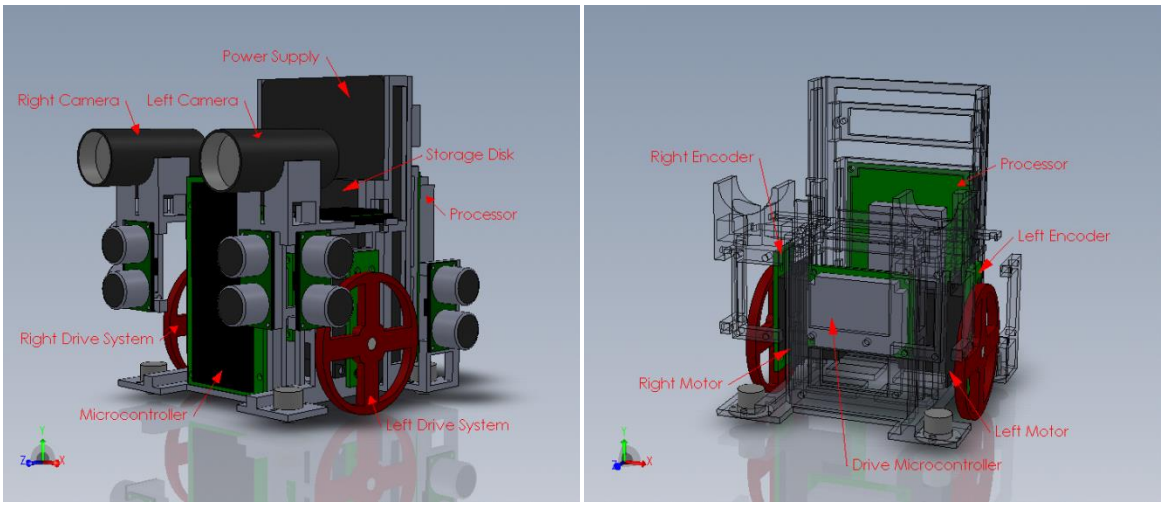


Figure 9-1 KCLBOT Dual Camera Stereo Vision Configuration

Using the KCLBOT model presented in Chapter 3, a new configuration was developed to support the stereo pair camera system. The new KCLBOT mobile robot stereo configuration capable of stereo vision is presented in Figure 9-1.

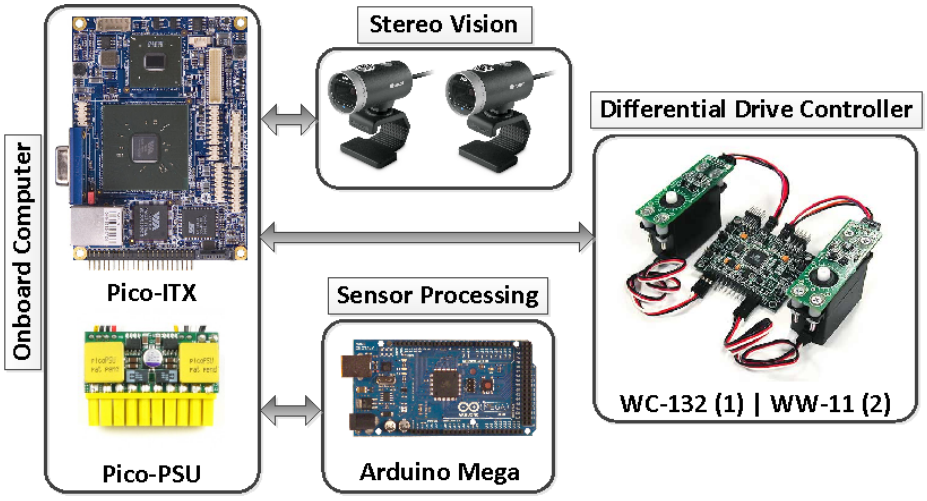


Figure 9-2 KCLBOT Dual Camera Electronic System

The KCLBOT is configured with Nubotic's WC-132 differential drive system [89], used to control the platform's navigation, Arduino's Mega 2560 microcontroller [90] with a Sparkfun Ethernet shield [91], used to process and communicate the platform's sensor data, and the stereo vision system, which uses two of Microsoft's HD cameras [92]. The central processing is done by VIA's Pico-ITX motherboard [93], with the data stored on a solid state disk and powered by a 6000mAh battery. A general overview of the stereo vision configured mobile robot is presented in Figure 9-1, with the electronic system presented in Figure 9-2.

9.3 Camera Geometry

The stereo configuration is composed of two identical cameras with matching focal lengths, aperture settings, shutter speed and ISO values. This is critical, as un-matched configurations will pose challenges to the stereo correspondence process.

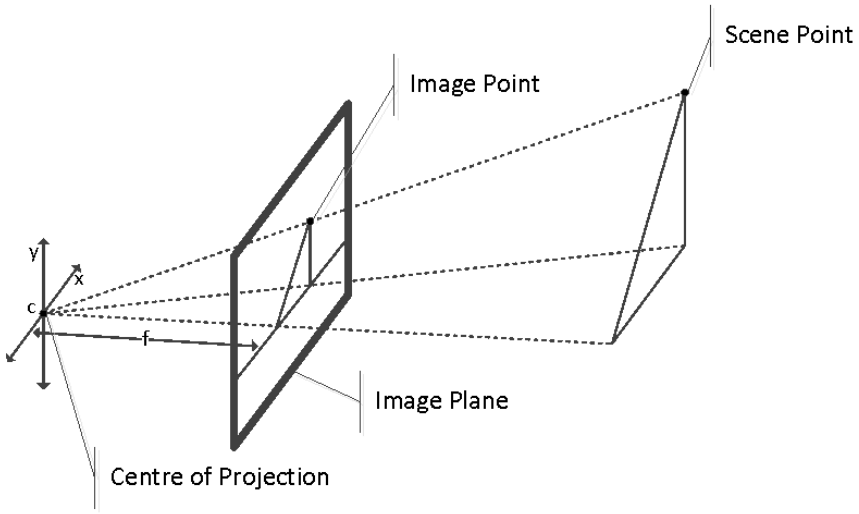


Figure 9-3 Camera Geometry Overview

When an image is captured, it is portrayed from the point of view of the image plane seen in Figure 9-3, where the image point is the translations from the centre of projection and the camera focal length f. The image point is two dimensional and our scene point is the corresponding 3-dimensional position of the object in the image.

$$\begin{bmatrix} x_p \\ y_p \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ w \end{bmatrix} \tag{9.1}$$

Where the scene points (x_s, y_s, z_s), and the image points (x_i, y_i) are represented by homogenous vectors in the linear transformation in equation (9.1). Where $x_i = x_p / \mathrm{w}$ and $y_i = y_p / \mathrm{w}$, where w is constant not assumed to be 1. This is representation of converting to homogenous image coordinates to the homogenous scene coordinates.

$$x_i = f\frac{x_s}{z_s} \tag{9.2}$$

$$y_i = f\frac{y_s}{z_s} \tag{9.3}$$

Where equations (9.2) and (9.3) describe the correspondence from the scene point to the image point based on multiplication of the focal length.

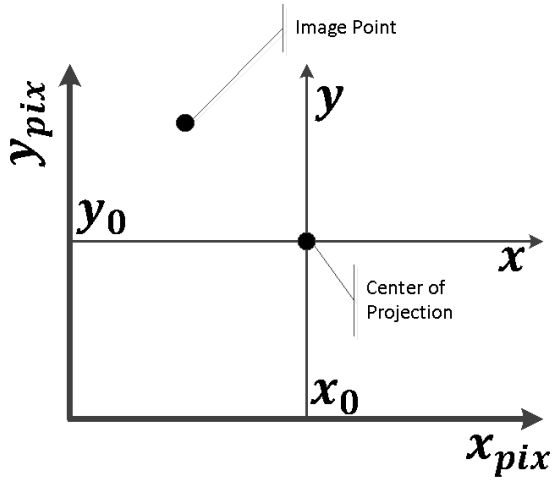


Figure 9-4 Camera Geometry – Image Plane

By transforming from length to pixel, the plane presented in Figure 9-4 is used as the orientation for the image view.

$$x_{pix} = \frac{(f.k_x + \mathrm{s})x_s}{z_s} + x_0 \ pixels \tag{9.4}$$

$$y_{pix} = \frac{f.k_y.y_s}{z_s} + y_0 \ pixels \tag{9.5}$$

Using the image plane model presented in Figure 9-4, the pixel coordinates are derived using equations (9.4) and (9.5), which describe the transformed pixel coordinates with respect to the centre of the projection, focal length and a scaling factor k. In addition equation (9.4), using the scaling variable s, takes into consideration any skewness that might occur.

$$\begin{bmatrix} x'_p \\ y'_p \\ w' \end{bmatrix} = \begin{bmatrix} f.k_x & s & x_0 & 0 \\ 0 & f.k_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ w \end{bmatrix} \quad (9.6)$$

Using the equations (9.4) and (9.5), and the knowledge that $x_{pix} = x'_p / w'$ and $y_{pix} = y'_p / w'$, a matrix is formed in equation (9.6), that conforms to structure set out in equation (9.1), which represents the transformation to pixels from the focal length in equation (9.1). It should be noted that $f, k_x, \mathrm{k}_y, x_s, \mathrm{y}_s, \mathrm{z}_s, x_0, \text{ and } y_0$ are measured in units of distance as millimetres. The variables x'_p and y'_p are measured as pixels. The methodology for deriving the intrinsic matrix in equation (9.6) comes from the work presented by Hartley and Zisserman [94].

9.4 Camera Calibration

Before using the camera captured images, the camera needs to be calibrated to correct for skewness, lens distortion, and to find the focal length and the coordinates of the centre of projection. These five coefficients are known as intrinsic parameters. The research provided by Tsai [95] and Zhang [96] offer techniques for solving the intrinsic parameters, starting by defining the homography matrix as follows:

$$H = \begin{bmatrix} P_0 & P_1 & P_2 \\ P_3 & P_4 & P_5 \\ P_6 & P_7 & P_8 \end{bmatrix} \tag{9.7}$$

Where H in equation (9.7) is the homography matrix [97] and describes the intrinsic parameters of equation (9.6). In the field of computer vision, any two images on the same planar surface in space are related by a homography. In order to solve the homography matrix, a dataset composed of correlated image and scene points is required. To achieve this, a reference in the scene plane is required, which is sufficiently easy to identify and map into image points. This is typically done using a chessboard type calibration image fixed onto a firm surface.

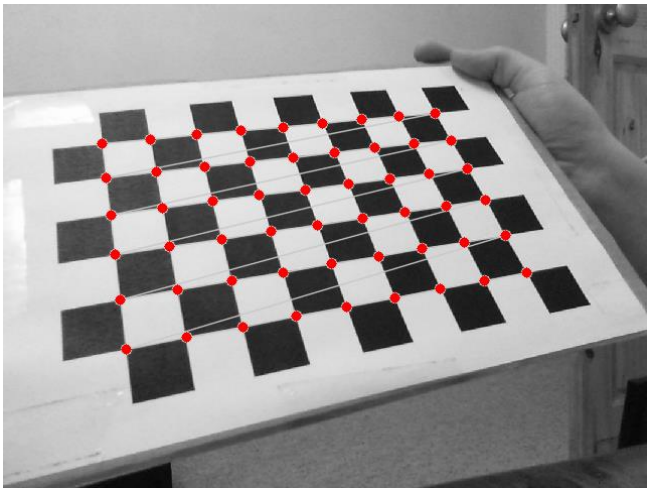


Figure 9-5 Chessboard Image Point Detection

Using a chessboard with a 9 corner width and a 6 corner height, as presented in Figure 9-5, the corners are identified using the Harris corner detection algorithm [98].

$$I_p = \begin{bmatrix} X_1 & Y_1 & W_1 & 0 & 0 & 0 & -x_{p1}X_1 & -x_{p1}Y_1 & -x_{p1}W_1 \\ 0 & 0 & 0 & X_1 & Y_1 & W_1 & -y_{p1}X_1 & -y_{p1}Y_1 & -y_{p1}W_1 \\ X_2 & Y_2 & W_2 & 0 & 0 & 0 & -x_{p2}X_2 & -x_{p2}Y_2 & -x_{p2}W_2 \\ 0 & 0 & 0 & X_2 & Y_2 & W_2 & -y_{p2}X_2 & -y_{p2}Y_2 & -y_{p2}W_2 \\ & & & & \vdots & & & & \\ X_n & Y_n & W_n & 0 & 0 & 0 & -x_{pn}X_n & -x_{pn}Y_n & -x_{pn}W_n \\ 0 & 0 & 0 & X_n & Y_n & W_n & -y_{pn}X_n & -y_{pn}Y_n & -y_{pn}W_n \end{bmatrix} \quad (9.8)$$

$$H' = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \end{bmatrix} \quad (9.9)$$

Where I_p from equation (9.8) describes the matrix of stored image point locations and H' from equation (9.9) is a manipulation of the homography matrix in equation (9.7), which is manipulated to help solve $I_p.H' = 0$.

Left Camera View

Right Camera View

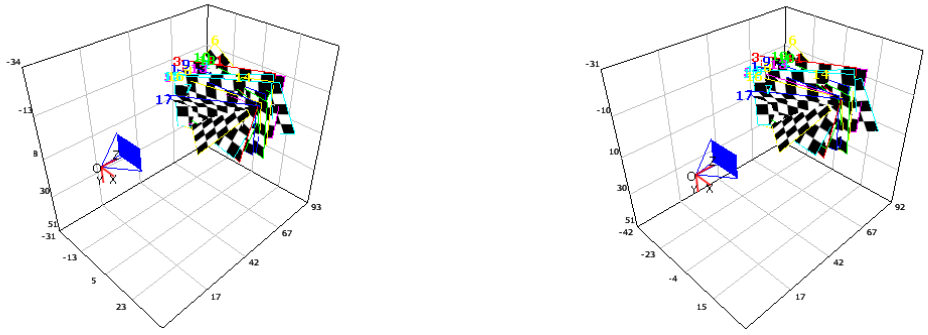


Figure 9-6 Experiment – Chessboard Configuration

After capturing a sufficient sample size of chessboard configurations, in different orientations as seen in Figure 9-6, a I_p matrix can be composed and the manipulated H' matrix can be solved using Levenberg-Marquardt algorithm [99] to return a numeric solution of the minimized error.

	$f.k_x$	$f.k_y$	s	x_0	y_0
Right Camera	590.17	589.65	0	323.39	242.88
Left Camera	590.76	589.99	0	313.96	247.09

Table 9-1 Experiment - Intrinsic Parameter Results

Using the solution to homography matrix, a set of experimental results of the intrinsic parameters are presented in Table 9-1. After finding the intrinsic parameters, the radial distortion of the camera needs to be rectified.

$$D_i = \begin{bmatrix} (\mathrm{x}_{pi} - x_{p0})(x_i^2 + y_i^2) & (\mathrm{x}_{pi} - x_{p0})(x_i^2 + y_i^2)^2 \\ (\mathrm{y}_{pi} - x_{p0})(x_i^2 + y_i^2) & (\mathrm{y}_{pi} - p_{p0})(x_i^2 + y_i^2)^2 \end{bmatrix} \tag{9.10}$$

$$d_i = \begin{bmatrix} (\breve{\mathrm{x}}_{pi} - x_{pi}) \\ (\breve{\mathrm{y}}_{pi} - y_{pi}) \end{bmatrix} \tag{9.11}$$

Where $(\mathrm{x}_p, \mathrm{y}_p)$ describes the ideal image coordinates and $(\breve{\mathrm{x}}_p, \breve{\mathrm{y}}_p)$ are the observed image coordinates. Using the methodology presented by Wang et al [100], the corresponding equations (9.10) and (9.11), can be used to solve the distortion matrix k using a linear least-squares method where $k = (D^T D)^{-1} D^T d$.

	k_1	k_2	k_3	k_4	k_5
Right Camera	0.0036	0.0831	0.0023	-0.0017	-0.2669
Left Camera	0.0018	-0.0009	0.0011	-0.0019	-0.0254

Table 9-2 Experiment – Distortion Coefficient Results

Instead of using a second order polynomial matrix, like in equation (9.10), a fifth order polynomial is used for higher accuracy and the experiment results of the distortion coefficients

are presented in Table 9-2. Due to the high quality of the cameras used for the experiment, the distortion coefficients are relatively low.

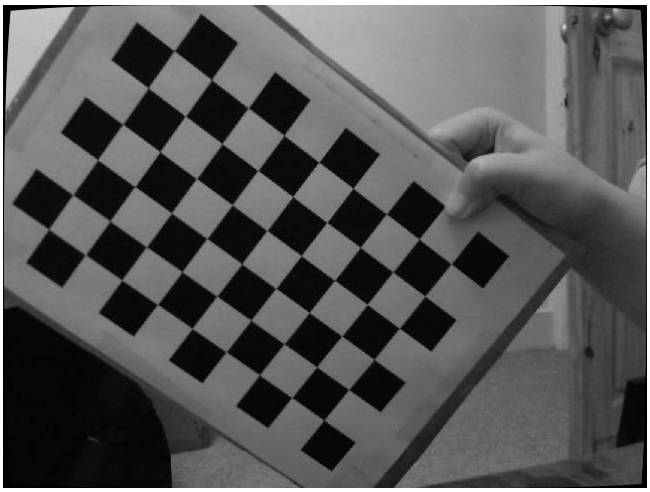


Figure 9-7 Experiment – Undistorted Right Camera Image

Using the distortion coefficients presented in Table 9-2, the radial distortion for the pin-hole camera can be removed from the image. An example of the image correction is presented in Figure 9-7, which corresponds to the right camera image, and the effects are notably visible on the image corners.

9.5 Extrinsic Parameters – Projection Matrix

Having solved the intrinsic parameters and found the distortion coefficients (see Table 9-1 and Table 9-2), the next step is to solve the extrinsic parameters.

$$\begin{bmatrix} x'_p \\ y'_p \\ w' \end{bmatrix} = A\begin{bmatrix} R & T \end{bmatrix}\begin{bmatrix} x_s \\ y_s \\ z_s \\ w \end{bmatrix} \tag{9.12}$$

Reforming equation (9.6) to take the extrinsic parameters into consideration, where A describes a matrix with the intrinsic parameters. The extrinsic parameters, denoted by R and T, are the coordinate transformations from a 3D world system to the 3D camera coordinate system. Where R is a rotation matrix and T describes the position of the origin of the world coordinate system, expressed in the coordinates of the cameras' coordinate system.

$$\begin{bmatrix} x'_{p(L,R)} \\ y'_{p(L,R)} \\ w'_{L,R} \end{bmatrix} = P_{L,R}\begin{bmatrix} x_s \\ y_s \\ z_s \\ w \end{bmatrix} \tag{9.13}$$

Where the intrinsic and extrinsic parameters are combined to produce a projection matrix P in equation (9.13), for both the right and left cameras.

$$\begin{bmatrix} X_1 & Y_1 & W_1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1W_1 \\ 0 & 0 & 0 & X_1 & Y_1 & W_1 & -v_1X_1 & -v_1Y_1 & -v_1W_1 \\ X_2 & Y_2 & W_2 & 0 & 0 & 0 & -u_2X_2 & -u_2Y_2 & -u_2W_2 \\ 0 & 0 & 0 & X_2 & Y_2 & W_2 & -v_2X_2 & -v_2Y_2 & -v_2W_2 \\ & & & & \vdots & & & & \\ X_n & Y_n & W_n & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nW_n \\ 0 & 0 & 0 & X_n & Y_n & W_n & -v_nX_n & -v_nY_n & -v_nW_n \end{bmatrix}\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = 0 \tag{9.14}$$

Where equation (9.13) is manipulated to produce equation (9.14), in the same way that the homography matrix was solved. The projection matrix P is transformed into a column vector of 12 values, where equation (9.14) can be simplified to $E_p.P = 0$.

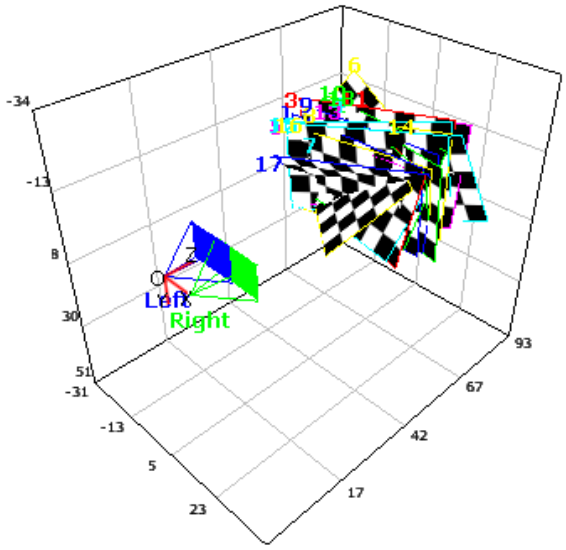


Figure 9-8 Experiment – Stereo Pair of Chess Board Configurations

After collecting a dataset of pairs of chessboard configurations, as depicted in Figure 9-8, the values are used to build the E_p from equation (9.14).

	P_L	P_R
P_{11}	577.1558	577.1558
P_{12}	0	0
P_{13}	231.0879	236.7010
P_{14}	0	816.1294
P_{21}	0	0
P_{22}	577.1558	577.1558
P_{23}	254.3942	254.3942
P_{24}	0	0
P_{31}	0	0
P_{32}	0	0
P_{33}	1	1
P_{34}	0	0

Table 9-3 Experiment - Extrinsic Parameter Results

Solving equation (9.14) using a singular value decomposition method [101], the experiment results of the individual projection matrices are presented in Table 9-3.

$$R = \begin{bmatrix} 0.9993 & -0.0282 & 0.0228 \\ 0.0228 & 0.9995 & -0.0073 \\ -0.0225 & 0.0079 & 0.9997 \end{bmatrix} \tag{9.15}$$

$$T = \begin{bmatrix} 1.3440 & 0.4127 & 0.1505 \end{bmatrix}^T \tag{9.16}$$

Using the experiment results from Table 9-3, the rotation and translation matrices from equation (9.13), which are presented in equations (9.15) and (9.16). Having both the intrinsic and extrinsic parameters, the model presented by equation (9.12) can be used to transform the image points from two images into a 3D world coordinate space.

9.6 Summary & Conclusions

In this chapter, a new KCLBOT configuration is presented, Figure 9-1, that is configured to hold a stereo vision capable system. The model to realize an object point in a three-dimensional space is derived by solving the projection matrix with the model's intrinsic and extrinsic parameters, while also correcting for skewness and radial distortion. The following research outcomes were presented in this chapter:

- A novel stereo vision capable mobile robot platform based on the KCLBOT was developed.
- A method to correct the skewness and radial distortion in images using a homography matrix and intrinsic parameter is derived.
- The solution to the projection matrix is derived enabling the transformation from image point to object point, ultimately enabling stereo vision for the mobile robot.

The research in this chapter is a contribution to researchers interested in developing mobile robot platforms capable of stereo vision. The KCLBOT mobile robot configuration is a valuable template for small mobile robots with cutting edge technology. The methodology to derive the 3D position of an object from two stereo paired images is a significant contribution to the computer vision research community.

A new mobile robot configuration has been presented with some very advanced features, which include a stereo vision system that can be used to build environment maps to assist with critical navigation tasks. A model was developed and derived to move from the image point to the object point using a projection matrix. This would not be possible without calibrating the camera's output images first, which is achieved by implementing a homography matrix to solve the possibility of image skewness and radial distortion. This is early research into stereo vision methods, but the initial results of the projection matrix look promising.

The research in this chapter is still in the early stages of its development and needs more time to develop. The following challenges have evolved from this early research. The image acquisition while the mobile robot has been moving has resulted in images that were un-

focused and as such, stereo correspondence was not possible. The methods described in this chapter will only perform without error when images are acquired while the cameras are stationary and without vibration. The auto focusing and white balance correction features in the cameras have led to erroneous results when their variables have not been matched.

Much further work is required to develop a stable stereo vision system for environment map building. The next stage of research needs to go towards developing a method for stereo correspondence to produce disparity maps, and exploratory research into these methods is critical. The research by Hasan et al [102] presents a viable solution to disparity map building and will be followed with future work. Also, more research is required to synchronise the camera capture parameters as mismatched parameters lead to erroneous results. Finally, the camera system requires a stabilization system or it will not be possible to use it when the mobile robot is moving. As this thesis already has presented research in a 9-axis self-localization method in Chapter 6, this sensor can be exploited to build a stabilizer to keep the camera level at all times.

Chapter 10 Summary & Conclusions

10.1 Research Outcomes & Benefits of the Research

The overall objective of this thesis was to deliver research on the self-localization method and environment map building tools for small form factor two-wheel manoeuvrable mobile robots, while also delivering research towards building a mobile robot of this classification, overcoming the mechanical, electrical, and software related challenges. Overall the outcome of the research has produced a technologically cutting edge mobile robot that has sold more than 20 units worldwide already. The researcher produced 12 first author publications [8] to [19] in three major robotics domains covering mobile robot modelling, self-localization methods and computer vision. The research covers a spectrum of robotics leading to the production of a mobile robot that surpasses mechanical, electrical and software challenges.

Mobile robot are very important to our daily lives, they have the capabilities to save lives, influence economic change by improving productivity, and help the human race explore uncharted wonders beyond the earth. Chapter 2 explores the background research into mechanical methods adopted for dead-reckoning in mobile robots and other relative localization methods. It explored various sensors used for inertial navigation, including the accelerometer and the gyroscope. Finally, the chapter concluded with a review of computer vision based methods for environment map building.

Chapter 3 presented the novel design for a small form factor mobile robot, nicknamed the KCLBOT, with a cutting edge fusion of relative and absolute sensor configurations, from ultra-sound range sensors to a 9-axis inertial measurement unit and quadrature shaft encoders. This thesis presented a new novel Arduino compatible circuit board which has built-in wireless compatibility and a single cell lithium polymer battery power management, none of which currently exist in the commercial market. The software for the mobile robot was developed on the Arduino IDE and the controller software was produced on an iOS platform, giving the mobile robot tele-operating ability from a mobile phone.

The research in Chapter 4 presented the constraints and singularities of a two wheel manoeuvrable mobile robot. The chapter derived the kinematic and dynamic model that described the behaviour of the two-wheel manoeuvrable mobile robot using Lagrangian of the known system which was further optimized using D'Alembert principle with Lagrange multipliers. Having derived the optimized behaviour of mobile robot, a closed-loop controller was derived to control the navigation of the platform from point A to point B using linear feedback. The chapter concluded with experiment simulations validating the behaviour of the model and the optimized PD controller.

Self-localization is a critical component in robotics and an essential feature for mobile robots. In view of this, Chapter 5 presented a novel self-localization method, exploiting the hybrid between the relative and absolute localization methods. The chapter derived the model for the distribution of power to the mobile robot motors, and derived the model for self-localizing the mobile robot with quadrature wheel encoders. Using the model for calculating distance travelled from the wheel encoders, the limitations of employing a numerical solution on the single compass self-localization method, which is based on a 6-bar mechanism, were evident. To overcome the limitation of the single compass method, a double compass approach is derived to solve the limitations and produce an analytical solution. The novel double compass configuration uses two digital compasses and quadrature shaft encoders to form an analytical solution to the mobile robot's pose and orientation. This outcome for localization is a big leap forward for small mobile robots. Chapter 5 concluded with a detailed statistical analysis showing a median error of less than 20mm when compared to an overhead marker tracking system.

The critical limitation in Chapter 4 and Chapter 5 was that both the optimized closed-loop PD controller and the novel self-localization for linear feedback into the controller do not consider slip. The research in Chapter 6 presented a novel self-localization method that gave the KCLBOT the ability to act as a self-balancing mobile robot, eliminating the need for the caster wheels and the attempt to prevent slip in the wheels. Chapter 6 continued the research into self-

localization but with particular focus on an absolute localization method using a trio of sensors to build an inertial measurement unit capable of 9DOF. The sensor configuration was based on an accelerometer, a magnetometer and a gyroscope sensor. The challenge for any inertial measurement unit is the computation rate and accuracy of the localization result. To overcome the heavy calculation rate of the Kalman filter method or the low accuracy of using a complementary filter, a new and novel method was developed which was based on a directional cosine matrix to solve for orientation with a PI controller. The chapter concluded with a detailed statistical analysis of the DCM method compared to the 3D glyph marker tracking system, showing a median error rate as low as 0.03 degrees.

Validating the results from Chapter 4 and Chapter 5, by employing an overhead camera system, presented the computational challenge of processing large amounts of images. Chapter 7 explored a method inspired by skip-list algorithm to improve the performance of a systematic searching of an image for non-natural markers. The chapter presented a pseudo random algorithm that, on its own, improves the search time by more than half a second compared to the systematic approach. In addition to the pseudo random algorithm, gradient policy is also applied to further improve the search time for the non-natural markers by estimating the location of the mobile robot based on its current and previous locations.

After self-localization, navigation environment map building is the next critical component for mobile robots. Without knowledge of the navigation environment, the mobile robot is blind to obstacles and would find it difficult to traverse any unknown environment. Chapter 8 presented the KCLBOT with a RGB-D sensor capable of building depth maps for environment map building. In this chapter, a model using the machine learning method was presented that transformed the RGB-D sensor data into a usable estimate for a measurable distance. Then, a novel model for exploiting the RGB-D sensor was presented with a model for converting the depth map information from the sensor into two-dimensional overview maps for the mobile robot. The statistical analysis showed the accuracy to be very impressive with a mean error of about ±0.15mm.

Chapter 8 presented limitations that required consideration. As the RGB-D sensor is not designed specifically for small mobile robots; it presented the challenges of size, weight, power consumption and computational requirements. These limitations validated the necessity for further research into alternative methods for building depth maps that can be used for environment map building. Continuing on the topic of environment map building, the early work into stereo vision was presented in Chapter 9. The chapter presented the challenges in setting up a stereo vision system for mobile robots, from the challenges of camera calibration to defining the intrinsic and extrinsic parameters to building a stereo depth view using two cameras. The results are limited to presenting the models for depth computation. Though more research is required in this area, the stereo vision method can be applied to environment maps.

In summary, the thesis produced the following major research contributions:

- In Chapter 3, a novel small form factor two-wheel manoeuvrable mobile robot with cutting edge technology for self-localization research was presented.
- In Chapter 4, an optimized closed-loop PD controller based on the Lagrangian of the two-wheel manoeuvrable mobile robot system was derived.
- In Chapter 5, a novel self-localization method was derived, which was based on the hybrid of quadrature wheel encoders and a dual compass configuration, returning an analytical solution for the mobile robot pose and orientation.
- In Chapter 6, a novel 9-axis sensor was presented with a closed-loop PI controller for three-dimensional orientation feedback, using a directional cosine matrix.
- In Chapter 7, a pseudo random algorithm with a gradient policy was derived to improve the computation time for tracking non-natural markers.
- In Chapter 8, a machine learning model was derived to analytically solve the transformation from RGB-D data to a depth estimation, which was used to produce two-dimensional environment maps.
- In Chapter 9, the research into stereo vision produced a solution to the projection matrix to move from a 2D image point to a 3D object point.

10.2 Limitations of the Current Research

It is a considerable challenge to develop a wirelessly controllable mobile robot platform that enables research methods into novel self-localization principles and environment map building. Chapter 3 presented the three major challenges in implementing a mobile robot of this classification. The first challenge was the mechanical design of the two-wheel manoeuvrable mobile robot to support and host all the electrical technology and to be functional as a manoeuvrable mobile robot. Secondly, the electrical challenge exists to implement a microcontroller based system that supported all the self-localization sensors inputs, power management, wireless communication, and motor control. Finally, implementing software at a firmware level to carry out data acquisition from the sensors, to managing the outputs to the mobile robot's motors, managing the system's power distribution and the bi-directional wireless communication presented a further challenge.

In Chapter 4, an optimized closed-loop PD controller was derived for the two-wheel manoeuvrable mobile robot, which was based on the holonomic and nonholonomic constraints of the system. The system is modelled on the Lagrangian of the known system and optimized with Lagrange multipliers. This model is limited to two-wheel manoeuvrable mobile robots only and centralized centre of gravity. It is also important to note that neither the model nor the controller take slip estimation into consideration.

In Chapter 5, a novel method for self-localization was derived to improve the linear feedback into the optimized PD controller from Chapter 4. The self-localization method was based on the hybrid of quadrature wheel encoders and a dual double configuration, which were used to form an analytical solution from a 6-bar mechanism model. Owing to the mechanical nature of the wheel encoders, the system was dependent on high resolution encoders because at small movement, low resolution encoders do not return any movement which ultimately leads to erroneous return. The magnetometers are also biased to magnetic interference and need to be calibrated for hard and iron compensation. Finally, in Chapter 4, this derived solution for self-localization assumed the mobile robot is free from wheel slip.

Considering the issue of wheel slip from the closed-loop optimized PD controller in Chapter 4 and the novel self-localization method from Chapter 5, a self-localization method to track the mobile robot in three-dimensional spaces was investigated. In Chapter 6, this was achieved using the combination of an accelerometer, a magnetometer, and a gyroscope to form an inertial measurement unit. Using the sensor data from the IMU system, a directional cosine matrix based PI controller was derived to feedback correction data to enable the mobile robot to function as a self-balancing robot and assist in preventing wheel slip. For the IMU to work, the three sensors needed to be calibrated.

In Chapter 7, a pseudo random algorithm with a gradient policy, inspired by a skip-list method, was derived to improve the non-natural marker tracking process. The accuracy of the method was limited by the resolution of overhead cameras and the size of the non-natural marker.

In Chapter 8, a machine learning base method was derived to transform RGB-D sensor data into depth information and used to build an environment map. The sensor was dependant on a well-lit environment and the sensor could only perform distance estimations on solid surfaces, as transparent surfaces return erroneous results. The distance estimation was limited between the ranges of 800mm to 3000mm, which is not ideal for small mobile robots. The sensor also proved to be too large for the KCLBOT, consumed a lot of power, and was computationally intensive.

Using the limitation presented in Chapter 8 about the RGB-D sensor, Chapter 9 carried out early research into stereo vision for small mobile robots. While this system solves the issues of power consumption, weight and form factor, the method is still computationally expensive.

10.3 Recommendations & Future Work

The research in Chapter 4 produced a functional optimized closed-loop PD controller, but further research is still required to improve the linear feedback into the controller. The most important limitation is the consideration of wheel slip.

In Chapter 5, a novel method for self-localization was presented using a combination of relative and absolute localization sensors. While the results were favourable for this method, further research into slip estimation is required, which might include the addition of an optical flow sensor to measure wheel slip or it might even be possible to measure wheel slip using the 9-axis IMU sensor from Chapter 6.

The novel 9-axis IMU sensor with the optimized closed-loop PI controller, from Chapter 7, produced high accurate self-localization results. It will be valuable to the research community to investigate how this novel method performs against other existing methods like the Kalman filter or the complementary filter.

The RGB-D sensor from Chapter 8 produced extremely accurate depth estimation results but was limited by a minimum range of 800mm. Further research into manipulating the focal lens of the camera system has the potential to improve the minimum range of the sensor. With release of the new sensor by PrimeSense and the advanced in lens optic better performance can be expected in the future.

The research in Chapter 9 is limited to the modelling of the projection matrix, the next step will be to use the projection matrix result to perform stereo correspondence and build disparity maps, which were outside of the scope of this thesis. Methods into improving the calibration process and reduction in computational requirements will have a benefit to the stereo vision research domain.

Chapter 11 References

[1] Leonard, J. J., and Durrant-Whyte, H. F., "Application of multi-target tracking to sonar-based mobile robot navigation," Proc. Decision and Control, 1990., Proceedings of the 29th IEEE Conference on, pp. 3118-3123 vol.3116.

[2] Voth, D., 2005, "Segway to the future [autonomous mobile robot]," Intelligent Systems, IEEE, 20(3), pp. 5-8.

[3] Brooks, R. A., 1987, "Intelligence without representation," Artificial Intelligence, 47, pp. 139–159.

[4] Peng, Y., Zhen, S., and Ciyu, J., "Relative Location Technology Based on Dead Reckoning and Ultrasonic Data Fusion," Proc. Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, pp. 255-258.

[5] Arsenio, A., and Ribeiro, M. I., "Absolute localization of mobile robots using natural landmarks," Proc. Electronics, Circuits and Systems, 1998 IEEE International Conference on, pp. 483-486 vol.482.

[6] Borenstein, J., Everett, H. R., Feng, L., and Wehe, D., 1997, "Mobile Robot Positioning & Sensors and Techniques," Journal of Robotic Systems, Special Issue on Mobile Robots., 14(4), pp. 231 – 249.

[7] GEORGIOU, E., AL-MILLI, S., DAI, J. S., ALTHOEFER, K., CHHANIYARA, S., and SENEVIRATNE, L., "International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)," Proc. Advances in Mobile Robotics, pp. 1106-1113.

[8] Georgiou, E., Dai, J., "Lagrangian D'Alembert Modeled Maneuverable Autonomous Non-Holonomic Mobile Robot Using a Closed Loop PD Controller," Proc. ASME/IEEE 2009 International Conference on Mechatronic and Embedded Systems and Applications, pp. 341-346.

[9] Georgiou, E., Jian, D., Luck, M., "Regularized Linear Regression for Distance Estimation with an RGB-D Sensor," Proc. International Conference on Autonomous Robot Systems and Competitions, pp. 87-91.

[10] Georgiou, E., and Jian, D., "Self-localization of an autonomous maneuverable nonholonomic mobile robot using a hybrid double-compass configuration," Proc. Mechatronics and its Applications (ISMA), 2010 7th International Symposium on, pp. 1-8.

[11] Georgiou, E., and Jian, D., "Using a dual compass configuration with shaft encoders for self-localization of an autonomous maneuverable nonholonomic mobile robot," Proc. Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on, pp. 142-149.

[12] Evangelos Georgiou, J. S. D. a. M. L., "The KCLBOT: A Double Compass Self-Localizing Maneuverable Mobile Robot," Proc. ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 427-435.

[13] Evangelos Georgiou, J. D. a. M. L., 2011, "The KCLBOT: A Framework of the Nonholonomic Mobile Robot Platform Using Double Compass Self-Localisation," Mobile Robots - Current Trends.

[14] Georgiou, E., Jian, D., Luck, M., "Self-localization using a 9DOF IMU Sensor with a Directional Cosine Matrix," Proc. Proceedings of the ASME 2013 International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE).

[15] Georgiou, E., and Jian, D., "Visual self-localization for nonholonomic mobile robots using a Hybrid Skip-list inspired Search Algorithm with a Gradient Policy," Proc. Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on, pp. 390-397.

[16] Georgiou, E., Jian, D., "Mobile Robot Environment Map Building Using a RGB-D Sensor," Proc. International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR), pp. 329-336.

[17] Evangelos Georgiou, J. D. a. M. L., 2011, "The KCLBOT: Exploiting RGB-D Sensor Inputs for Navigation Environment Building and Mobile Robot Localization," International Journal of Advanced Robotic Systems, 8(4), pp. 194-202.

[18] Georgiou, E., Dai, J., Luck, M., "A Stereo Vision Model for a Small Form Factor Autonomous Mobile Robot KCLBOT," Proc. International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, pp. 81-88.

[19] Evangelos Georgiou, J. S. D. a. M. L., "The KCLBOT: The Challenges of Stereo Vision for a Small Autonomous Mobile Robot," Proc. ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 807-813.

[20] Khan, M., 2013, "The Development of a Mobile Medical Robot Using ER1 Technology," Potentials, IEEE, 32(4), pp. 34-37.

[21] Kane, G., Boesecke, R., Raczkowsky, J., and Worn, H., "Kinematic path-following control of a mobile robot on arbitrary surface," Proc. Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1562-1567.

[22] Wilson, J. C., and Berardo, P. A., "Automatic inspection of hazardous materials by mobile robot," Proc. Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on, pp. 3280-3285 vol.3284.

[23] Changhwan, C., and SeungHo, J., "Pipe Inspection Robot with an Automatic Tracking System Using a Machine Vision," Proc. SICE-ICASE, 2006. International Joint Conference, pp. 1285-1290.

[24] Espinoza, J. L., Sanchez, A. L., and Osorio, M., "Exploring unknown environments with mobile robots using SRT-Radial," Proc. Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pp. 2089-2094.

[25] Dutton, B., and Maloney, E. S., 1985, Dutton's Navigation & piloting, Naval Institute Press.

[26] Dimmler, M., and Dayer, C., 1996, "Optical encoders for small drives," Mechatronics, IEEE/ASME Transactions on, 1(4), pp. 278-283.

[27] Nickson, P., 1985, "Solid-State Tachometry," Sensors, pp. 23-26.

[28] Everett, H. R., 1995, Sensors for mobile robots: theory and application, A.K. Peters.

[29] Klarer, P. R., 1988, "Simple 2-D Navigation for Wheeled Vehicles," Sandia Report SAND88-0540, Sandia National Laboratories.

[30] Crowley, J. L., and Reignier, P., 1993, "Asynchronous Control of Rotation and Translation for a Robot Vehicle," Robotics and Autonmous Systems, Vol(10).

[31] Parish, D. W., and Grabbe, R. D., "Robust exterior autonomous navigation," pp. 280-291.

[32] Liu, H. H. S., and Pang, G. K. H., 2001, "Accelerometer for mobile robot positioning," Industry Applications, IEEE Transactions on, 37(3), pp. 812-819.

[33] Hyun, M., Hyoung-Ki, L., Kiwan, C., SeokWon, B., YeunBae, K., and Sangryong, K., "Mobile Robot Localization Using a Gyroscope and Constrained Kalman Filter," Proc. SICE-ICASE, 2006. International Joint Conference, pp. 2098-2103.

[34] Kibler, S. G., Hauer, A. E., Giessel, D. S., Malveaux, C. S., and Raskovic, D., "IEEE Micromouse for mechatronics research and education," Proc. Mechatronics (ICM), 2011 IEEE International Conference on, pp. 887-892.

[35] Joblove, G., and Greenberg, D., "Color spaces for computer graphics," Proc. 5th Annual Conference on Computer Graphics and Interactive Techniques.

[36] Lee, S., Song, J., 2005, "Mobile Robot Localization using Range Sensors : Consecutive Scanning and Cooperative Scanning," International Journal of Control, Automation, and Systems, 3(1), pp. 1-14.

[37] Surmann, H., Nuchter, A., Hertzberg, J., 2003, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," Robotics and Autonomous Systems(45), pp. 181-198.

[38] Nishihara, H., 1984, "Practical real-time imaging stereo matcher," Optical Engineering, 23(5).

[39] Huber, E. K., D., "Using stereo vision to pursue moving agents with a mobile robot," Proc. IEEE Conference on Robotics and Automation, pp. 2340 - 2346.

[40] Haverinen, J., Parpala, M., Roning, J., "A Miniature Mobile Robot With a Color Stereo Camera System for Swarm Robotics Research," Proc. IEEE International Conference on Robotics and Automation, pp. 2483-2486.

[41] Scheuer, A., and Xie, M., "Continuous-curvature trajectory planning for manoeuvrable non-holonomic robots," Proc. Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on, pp. 1675-1680 vol.1673.

[42] Sangeun, C., and Newman, W. S., "Design and evaluation of a laser-cutting robot for laminated, solid freeform fabrication," Proc. Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on, pp. 1551-1556 vol.1552.

[43] Ruiz, A. P., and Meizel, D., "Obstacle modeling in a set membership for navigation systems based in ultrasonic range sensors," Proc. ISAI/IFIS 1996. Mexico-USA Collaboration in Intelligent Systems Technologies. Proceedings, pp. 85-92.

[44] Hori, Y., "Robust and adaptive control of a servomotor using low precision shaft encoder," Proc. Industrial Electronics, Control, and Instrumentation, 1993. Proceedings of the IECON '93., International Conference on, pp. 73-78 vol.71.

[45] KyuCheol, P., Hakyoung, C., Jongbin, C., and Jang Gyu, L., "Dead reckoning navigation for an autonomous mobile robot using a differential encoder and a gyroscope," Proc. Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference on, pp. 441-446.

[46] Denysyuk, P., and Teslyuk, T., "Main algorithm of mobile robot system based on the microcontroller Arduino," Proc. Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory (DIPED), 2013 XVIIIth International Seminar/Workshop on, pp. 209-212.

[47] Adrian Tigauan, E. L., 2012, "Wireless Remote Control System," Carpathian Journal of Electronic and Computer Engineering, 5, p. 4.

[48] Torresen, J., Hafting, Y., and Nymoen, K, "A new wi-fi based platform for wireless sensor data collection," Proc. International Conference on New Interfaces For Musical Expression, pp. 337–340.

[49] Adam Kaliszan, M. G., 2011, "A Didactic Platform for Practical Study of Real Time Embedded Operating Systems," The International Journal on Advances in Telecommunications, 4, pp. 293-314.

[50] V.A. Fletcher, D. A. C., "WaterLogged, an assistive device for Waka Ama," Proc. The 16th Electronics New Zealand Conference (ENZCon), pp. 125 - 130.

[51] Tejaswini. S, B. S. S., Prasanna kumar, B. G. Sudarshan, "Design of Inexpensive User Friendly Digital Vital Monitoring System " Proc. International Conference on Computer Science and Informatics (ICCSI), pp. 42 - 46.

[52] Lo, G., Gonzalez-Valenzuela, S., and Leung, V. C. M., 2013, "Wireless Body Area Network Node Localization Using Small-Scale Spatial Information," Biomedical and Health Informatics, IEEE Journal of, 17(3), pp. 715-726.

[53] Ahmed ElShafee, K. A. H., 2012, "Design and Implementation of a WiFi Based Home Automation System," World Academy of Science, Engineering and Technology, pp. 2177 - 2183.

[54] Kavya M.C, H. G., 2013, "Position measurement and control of a Piezoresistive Contactless position sensor through Ethernet," International Journal of Engineering Research & Technology (IJERT), 2(9), pp. 617 - 620.

[55] Noetic Design, I., "Nubotics," http://www.nubotics.com/.

[56] Dudek, G., and Jenkin, M., 2000, Computational principles of mobile robotics, Cambridge University Press.

[57] Kolmanovsky, I. a. M., N. , 1995, "Developments in nonholonomic control problems," IEEE Control Systems Magazine, pp. 20-36.

[58] Lew, A., Marsden, J., Ortiz, M. and West, M, 2004, "An Overview of Variational Integrators. In: Finite Element Methods: 1970's and Beyond. Theory and engineering applications of computational methods," International Center for Numerical Methods in Engineering (CIMNE), pp. 1-18.

[59] Tongying, G., Fengyan, H., Haichen, W., and Languang, Z., "Application of Monte Carlo Localization Algorithm on Mobile Robot," Proc. Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on, pp. 533-536.

[60] Fox, D., Burgard, W., and Thrun, S., 1999, "Markov localization for mobile robots in dynamic environments," Journal of Artificial Intelligence Research(11).

[61] Sooyong, L., and Jae-Bok, S., "Robust mobile robot localization using optical flow sensors and encoders," Proc. Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, pp. 1039-1044 Vol.1031.

[62] North, E., Georgy, J., Tarbouchi, M., Iqbal, U., and Noureldin, A., "Enhanced mobile robot outdoor localization using INS/GPS integration," Proc. Computer Engineering & Systems, 2009. ICCES 2009. International Conference on, pp. 127-132.

[63] Alexander, J. a. M., J. , 1989, "On the kinematics of wheeled mobile robots," The International Journal of Robotics Research, 8, pp. 15-27.

[64] Kluever, C. A., 2014, Dynamic Systems: Modeling, Simulation, and Control 1E Wiley E-Text Student Package, John Wiley & Sons, Incorporated.

[65] Hellmers, H., Norrdine, A., Blankenbach, J., and Eichhorn, A., "An IMU/magnetometer-based Indoor positioning system using Kalman filtering," Proc. Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on, pp. 1-9.

[66] Acharyya, S. a. M., M. , 2009, "Performance of EAs for four-bar linkage synthesis," Mechanism and Machine Theory, 44(9), pp. 1784-1794.

[67] Zhang, G., Wang, X., Liang, Y., and Li, J. ,, 2010, "Fast and Robust Spectrum Sensing via Kolmogorov-Smirnov Test," IEEE TRANSACTIONS ON COMMUNICATIONS, 58(12), pp. 3410-3416.

[68] Hu, L., Zhu, H.,, 2005, "Bounded Brownian bridge model for UWB indoor multipath channel," IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications,, pp. 1411-1414.

[69] Balakrishna, R., and Ghosal, A., 1995, "Modeling of slip for wheeled mobile robots," Robotics and Automation, IEEE Transactions on, 11(1), pp. 126-132.

[70] Qin, Y., Liu, Y., Zang, X., and Liu, j., "Balance control of two-wheeled self-balancing mobile robot based on TS fuzzy model," Proc. Strategic Technology (IFOST), 2011 6th International Forum on, pp. 406-409.

[71] Hadjili, M. L., and Kara, K., "Modelling and control using Takagi-Sugeno fuzzy models," Proc. Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International, pp. 1-6.

[72] Qian, H., Liping, C., Weiwei, Q., Peng, L., Songling, Y., and Qifang, L., "Controling simulation study on two-wheeled self-balancing electrical motorcycle based on ADAMS and MATLAB," Proc. Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011, pp. 1704-1707.

[73] Xiaoming, Z., and Lizhen, G., "A novel auto-calibration method of the vector magnetometer," Proc. Electronic Measurement & Instruments, 2009. ICEMI '09. 9th International Conference on, pp. 1-145-141-150.

[74] Seong Yun, C., and Chan Gook, P., 2003, "Tilt compensation algorithm for 2-axis magnetic compass," Electronics Letters, 39(22), pp. 1589-1590.

[75] Piovan, G., and Bullo, F., 2012, "On Coordinate-Free Rotation Decomposition: Euler Angles About Arbitrary Axes," Robotics, IEEE Transactions on, 28(3), pp. 728-733.

[76] Choukroun, D., Weiss, H., Bar-Itzhack, I. Y., and Oshman, Y., 2010, "Direction Cosine Matrix Estimation from Vector Observations using a Matrix Kalman Filter," Aerospace and Electronic Systems, IEEE Transactions on, 46(1), pp. 61-79.

[77] Mahony, R., Hamel, T., and Pflimlin, J. M., 2008, "Nonlinear Complementary Filters on the Special Orthogonal Group," Automatic Control, IEEE Transactions on, 53(5), pp. 1203-1218.

[78] Tar, J. K., Kozeowski, K., Patkai, B., and Tikk, D., "Convergence properties of the modified renormalization algorithm based adaptive control supported by ancillary methods," Proc. Robot Motion and Control, 2002. RoMoCo '02. Proceedings of the Third International Workshop on, pp. 51-56.

[79] Hincapie, M., Caponio, A., Rios, H., and Mendivil, E. G., "An introduction to Augmented Reality with applications in aeronautical maintenance," Proc. Transparent Optical Networks (ICTON), 2011 13th International Conference on, pp. 1-4.

[80] Powers, R. M., and Pao, L. Y., "Using Kolmogorov-Smirnov Tests to Detect Track-Loss in the Absence of Truth Data," Proc. Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on, pp. 3097-3104.

[81] Ferdinando, H., Khoswanto, H., and Purwanto, D., "Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATMega8535," Proc. Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on, pp. 1-5.

[82] Fourati, H., Manamanni, N., Afilal, L., and Handrich, Y., "Position estimation approach by Complementary Filter-aided IMU for indoor environment," Proc. Control Conference (ECC), 2013 European, pp. 4208-4213.

[83] Pugh, W., 1990, "Skip lists: a probabilistic alternative to balanced trees," Commun. ACM, 33(6), pp. 668-676.

[84] Kumar, T., and Verma, K., 2010, "A Theory Based on Conversion of RGB image to Gray image," International Journal of Computer Applications, 7(2), pp. 7-10.

[85] Herter, T., and Lott, K., 1993, "Algorithms for decomposing 3-D orthogonal matrices into primitive rotations," Journal on Computers & Graphics, 17(5), pp. 517-527.

[86] Hyunwoong, P., Sooyong, L., and Wan-Kyun, C., "Obstacle Detection and Feature Extraction using 2.5D Range Sensor System," Proc. SICE-ICASE, 2006. International Joint Conference, pp. 2000-2004.

[87] Tomono, M., "Environment modeling by a mobile robot with a laser range finder and a monocular camera," Proc. Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on, pp. 133-138.

[88] Zezhong, X., Jilin, L., Zhiyu, X., and Han, L., "Map building for indoor environment with laser range scanner," Proc. Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on, pp. 136-140.

[89] "WC-132 WheelCommander Motion Controller," http://www.nubotics.com/products/wc132/index.html

[90] Arduino, 2011, "Arduino Mega 2560," http://arduino.cc/en/Main/ArduinoBoardMega2560.

[91] Electronics, S., 2012, "Arduino Ethernet Shield," http://www.sparkfun.com/products/9026.

[92] Microsoft, 2011, "Microsoft LifeCam Cinema," http://www.microsoft.com/hardware/en-us/p/lifecam-cinema.

[93] Inc, V. T., "Artigo Pico-ITX," http://www.via.com.tw/en/initiatives/spearhead/pico-itx/index.jsp.

[94] Hartley, R. I., and Zisserman, A., 2004, Multiple View Geometry in Computer Vision, Cambridge University Press, ISBN: 0521540518.

[95] Tsai, R., 1987, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses," IEEE Journal of Robotics and Automation, 3(4), pp. 323-344.

[96] Zhang, Z., 1998, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), pp. 1330–1334.

[97] Okatani, T., and Deguchi, K., "Autocalibration of a projector-screen-camera system: theory and algorithm for screen-to-camera homography estimation," Proc. Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pp. 774-781 vol.772.

[98] Harris, C., Stephens, M., "A Combined Corner and Edge Detection," Proc. In Proceedings of The Fourth Alvey Vision Conference, pp. 147-151.

[99] Levenberg, K., 1944, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quarterly of Applied Mathematics, 2, pp. 164–168.

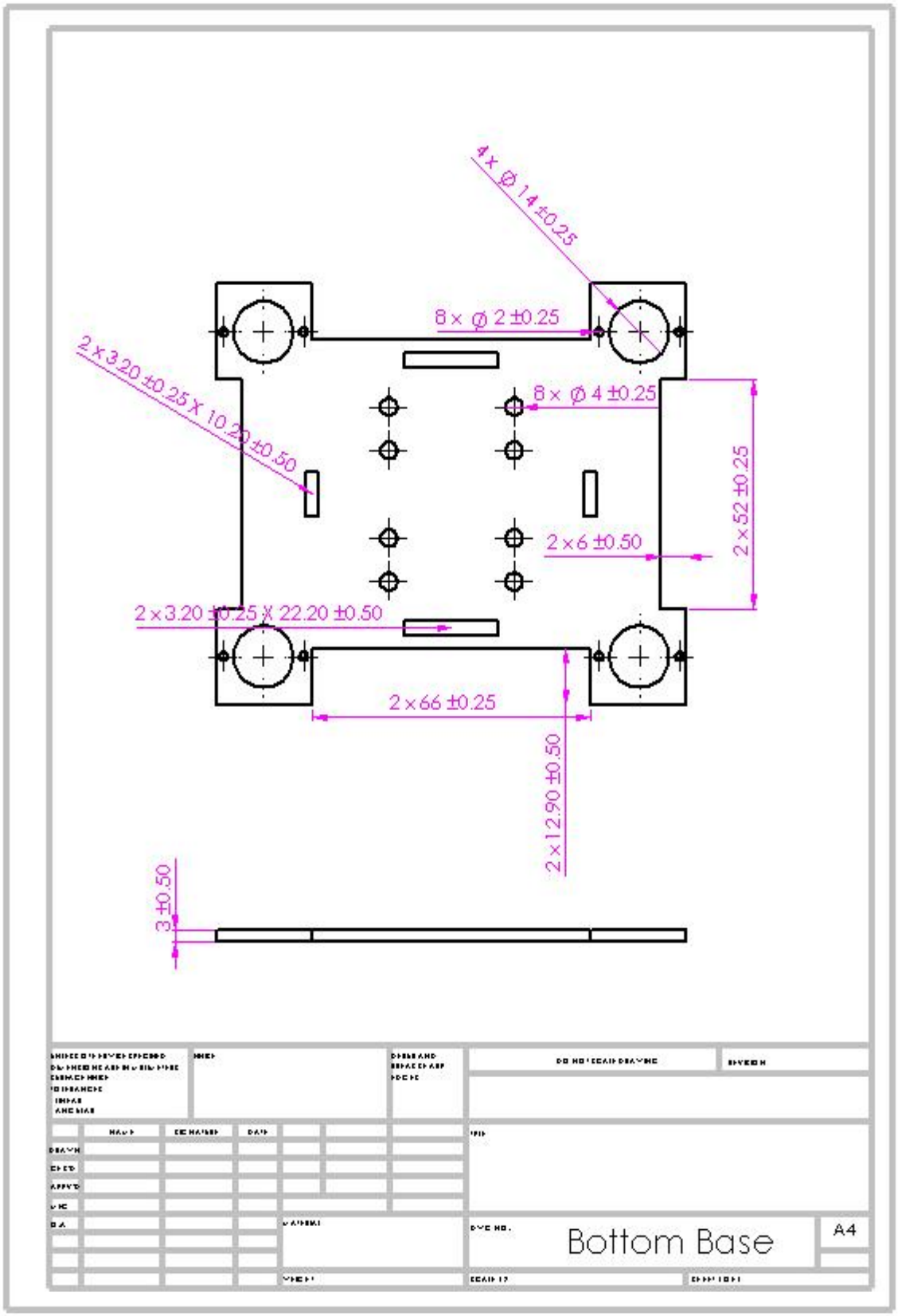
[100] Wang, J., Shi, F., Zhang, J., Liu, Y., 2008, "A new calibration model of camera lens distortion," Pattern Recognition, 41(2), pp. 607-615.

[101] Press, W., Teukolsky, S., Vetterling, W., Flannery, B. , 1992, Singular Value Decomposition, Cambridge University Press.

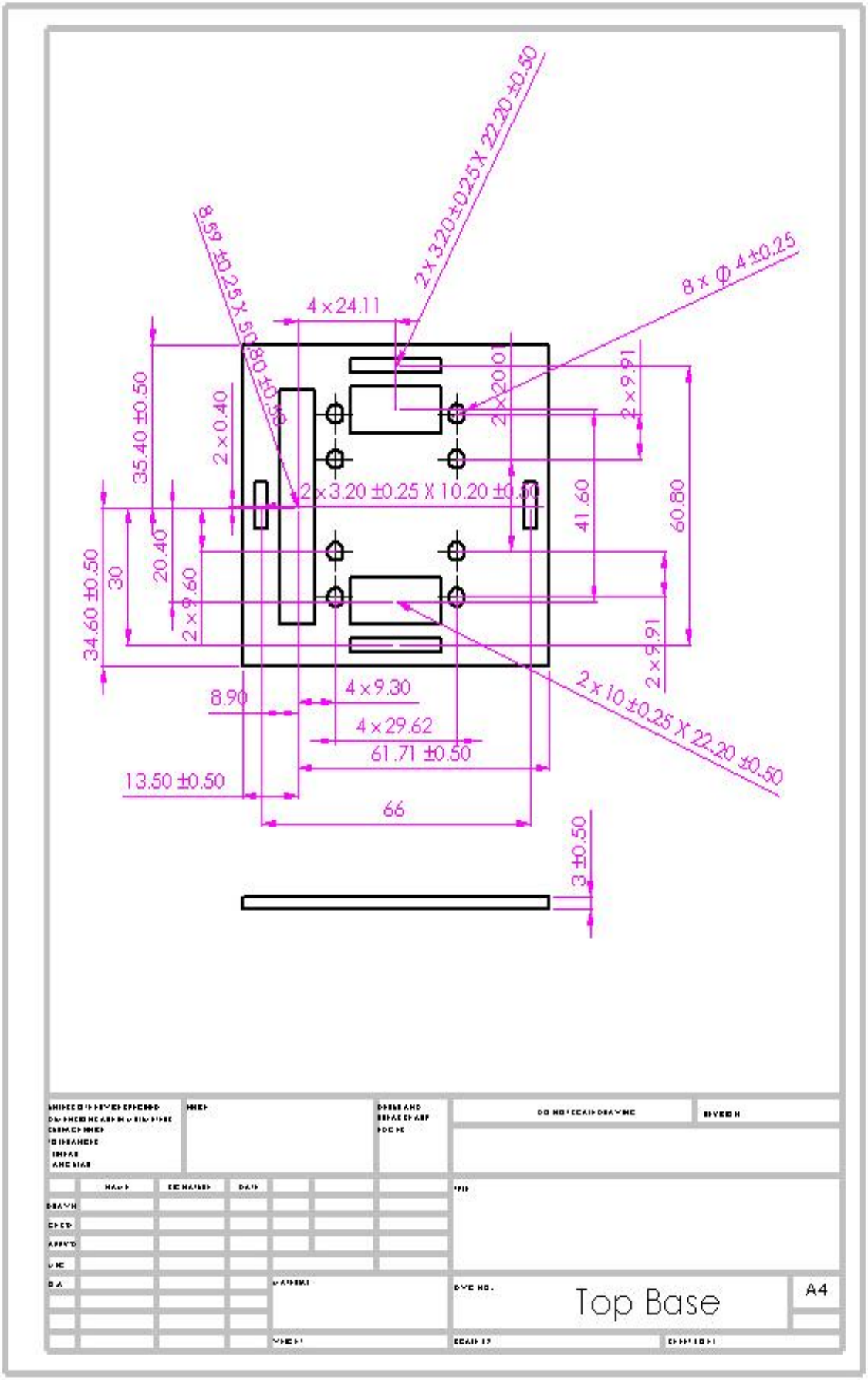
[102] Hasan, A. H. A., Hamzah, R. A., and Johar, M. H., "Disparity Mapping for Navigation of Stereo Vision Autonomous Guided Vehicle," Proc. Soft Computing and Pattern Recognition, 2009. SOCPAR '09. International Conference of, pp. 575-579.

Chapter 12 Appendix

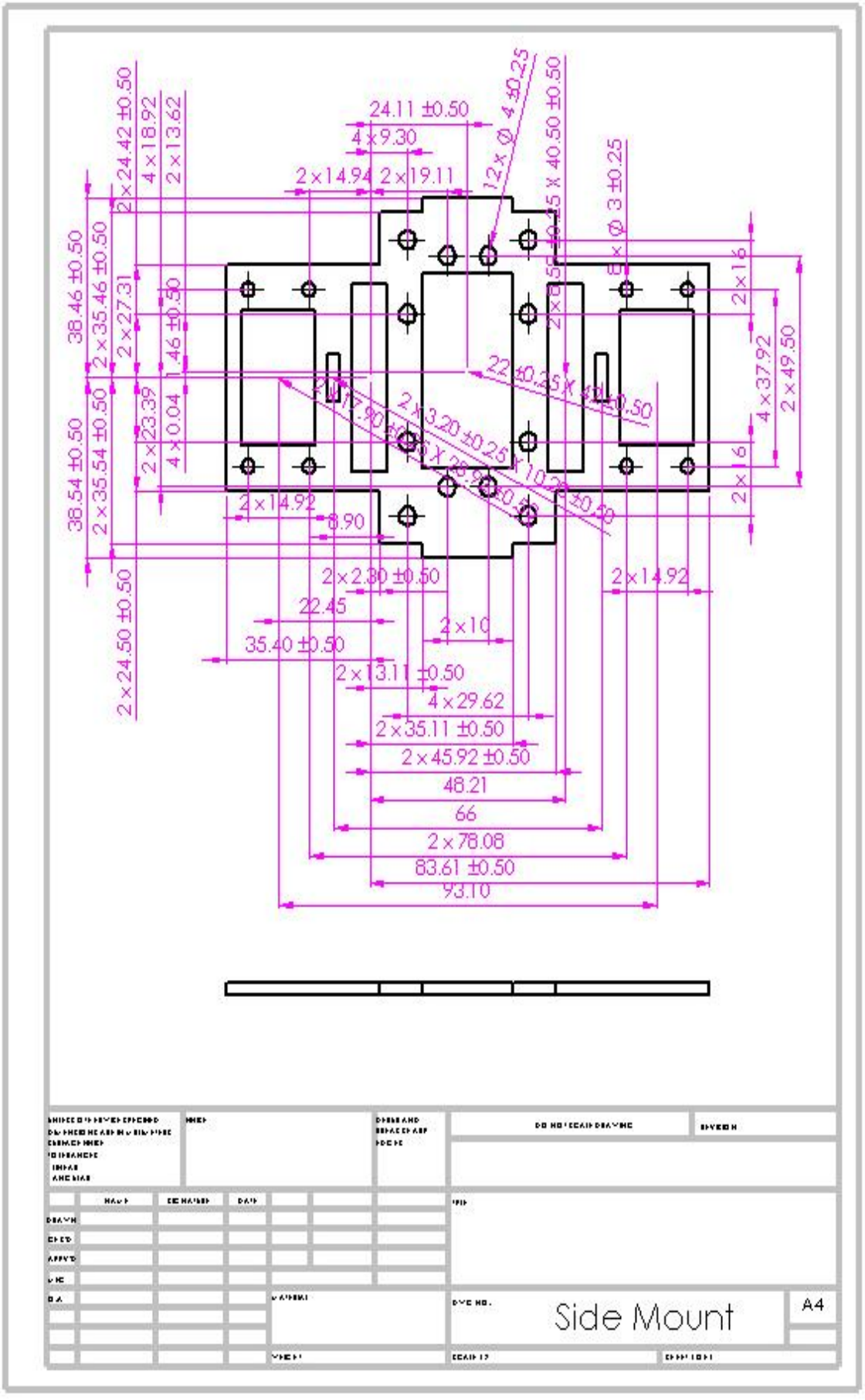
12.1 KCLBOT Mechanical Designs



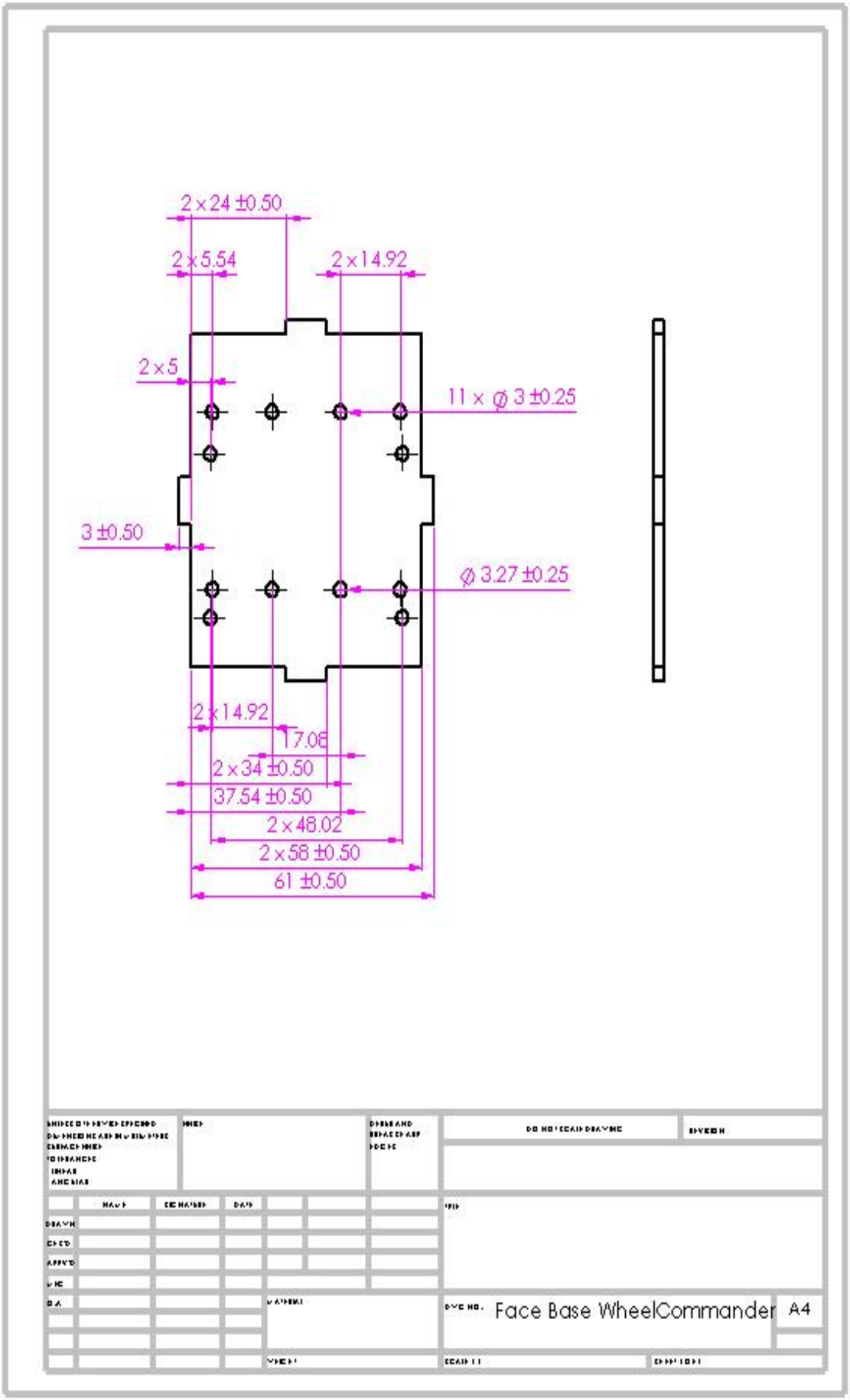
KCLBOT Designs 1 Mobile Robot Bottom Base



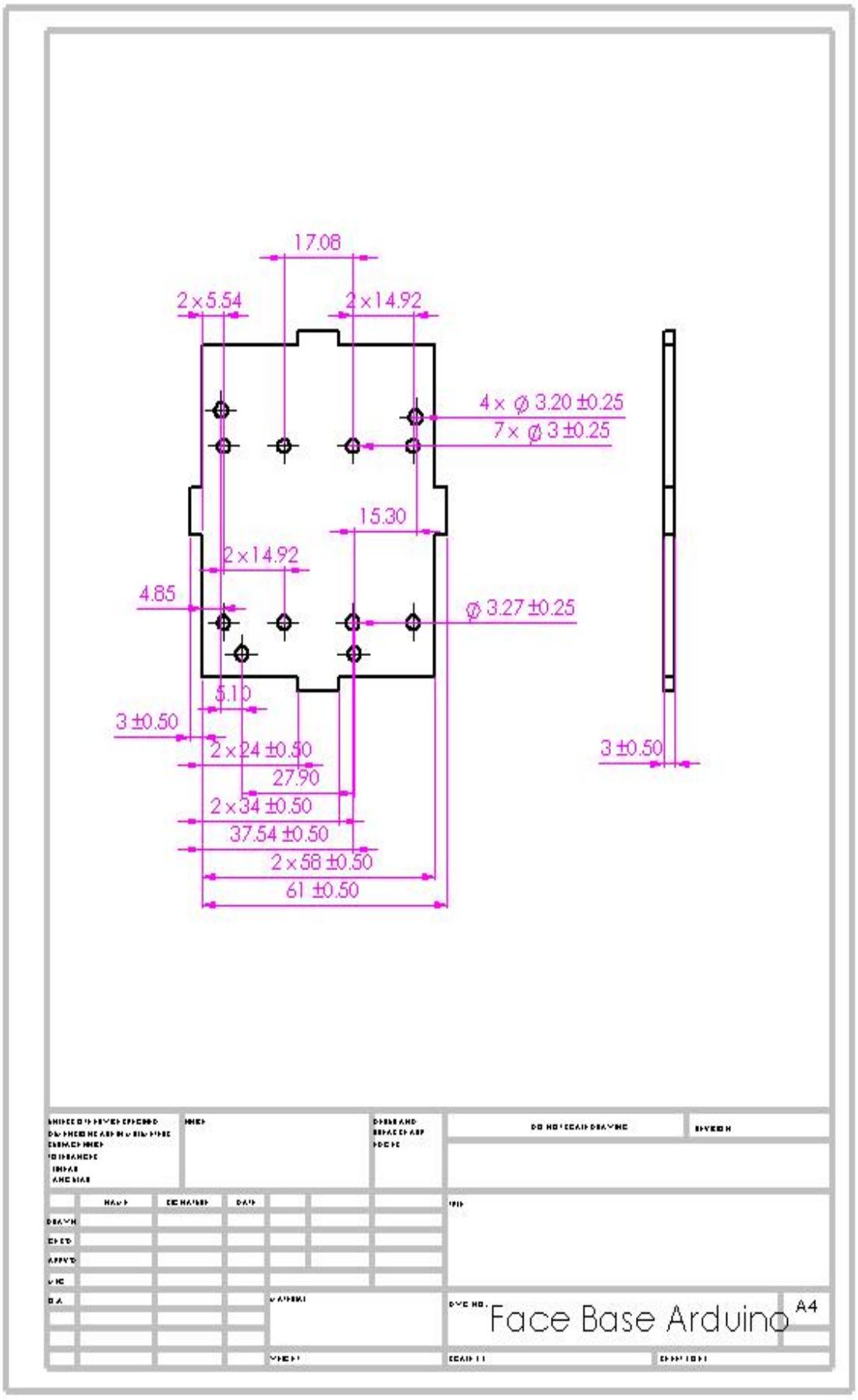
KCLBOT Designs 2 Mobile Robot Top Base



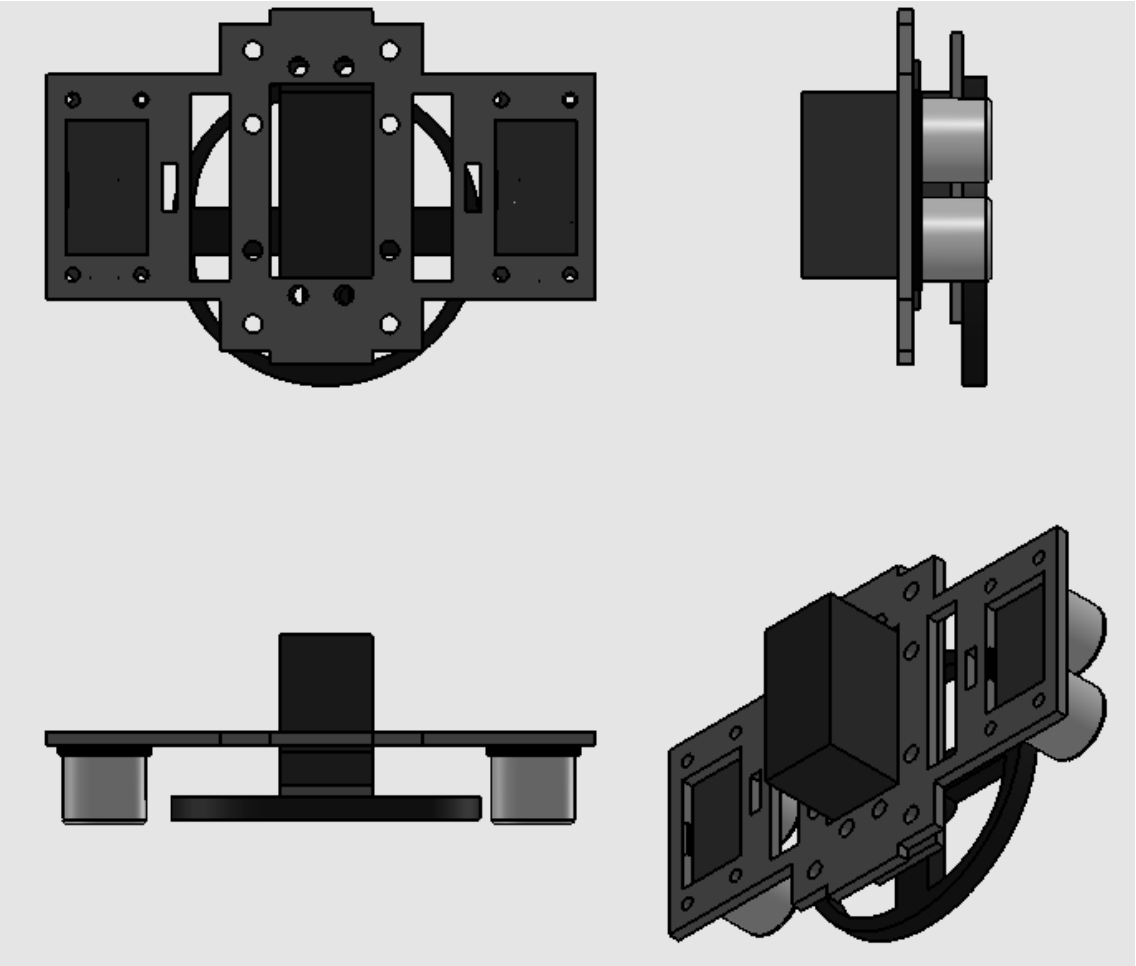
KCLBOT Designs 3 Mobile Robot Side Mount



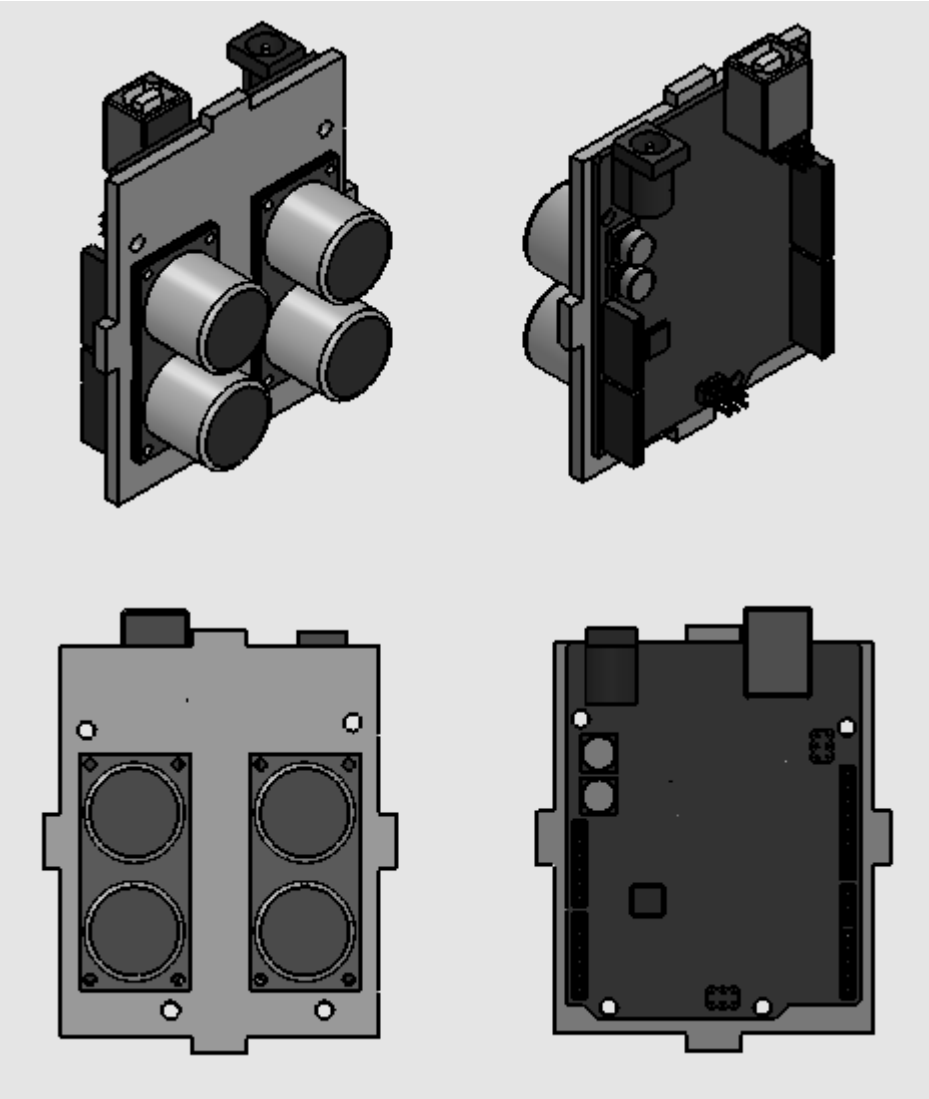
KCLBOT Designs 4 Mobile Robot Face Base for Wheel Commander



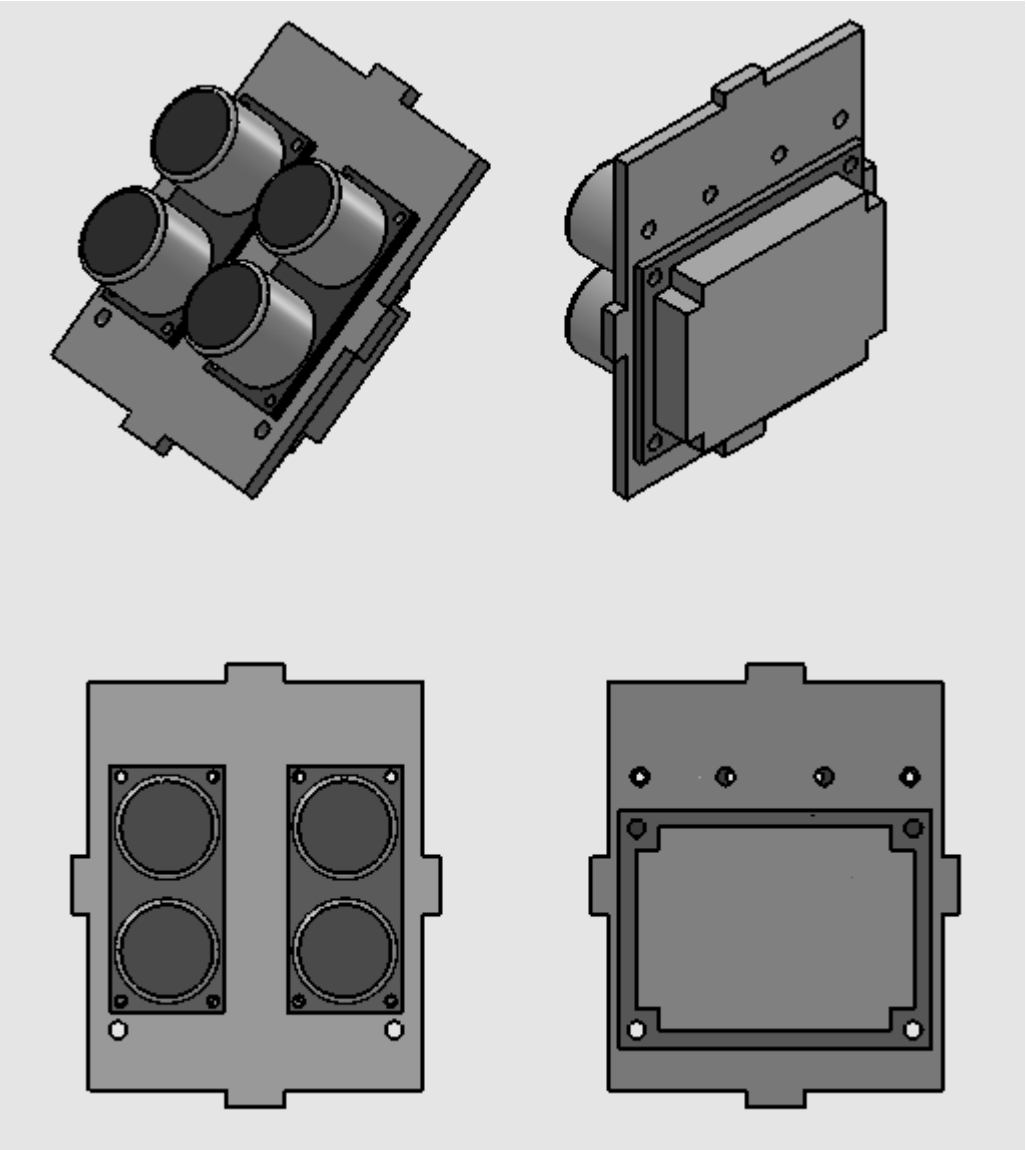
KCLBOT Designs 5 Mobile Robot Face Base for Arduino Board



KCLBOT Designs 6 Mobile Robot Side View

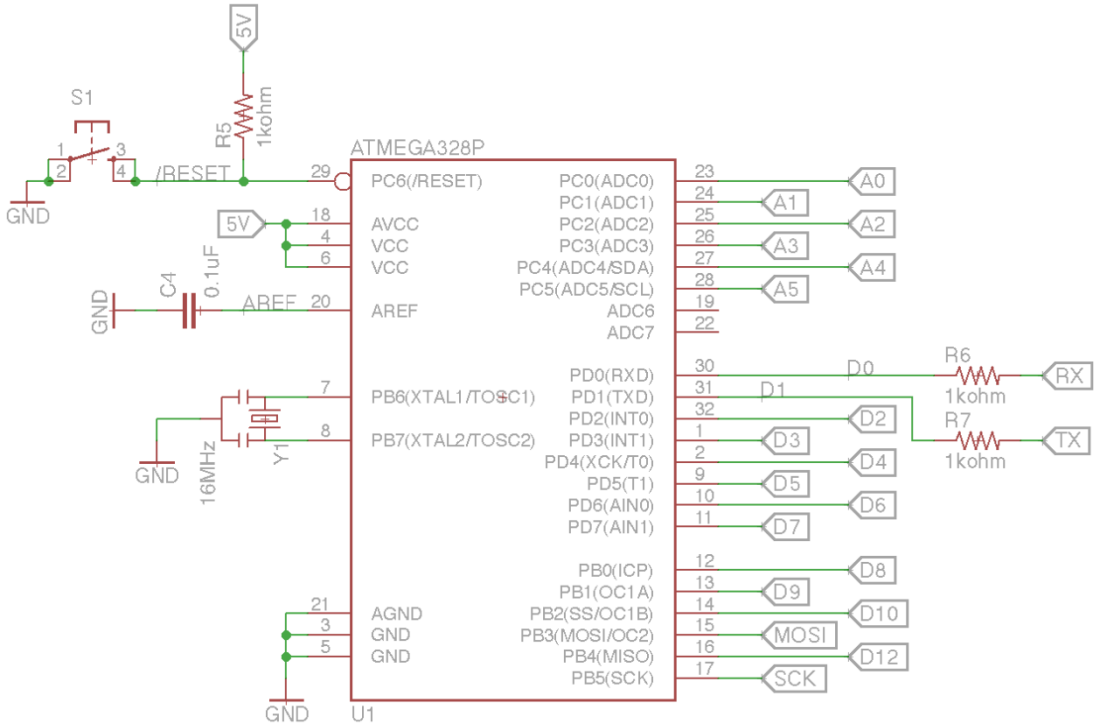


KCLBOT Designs 7 Arduino Controller Mount

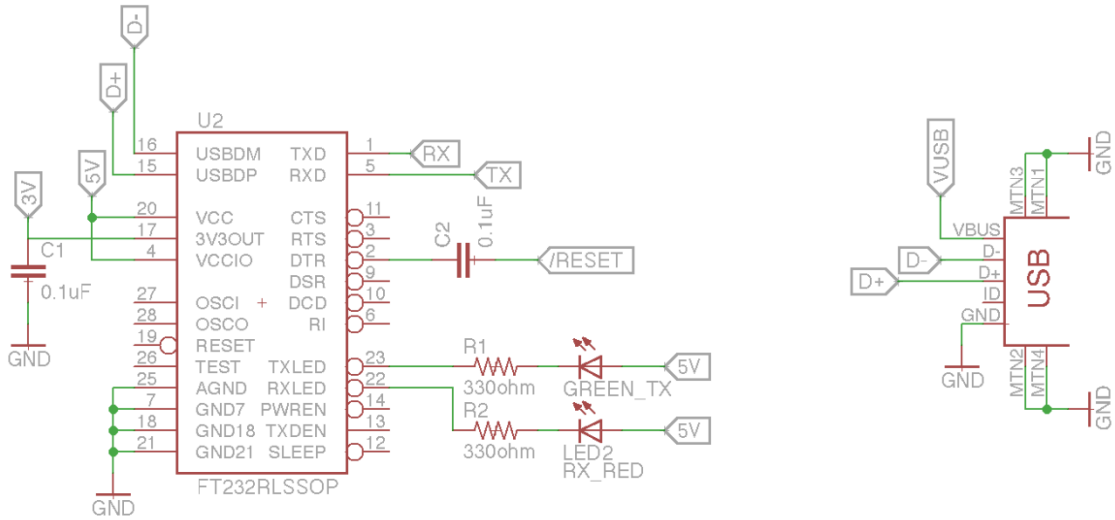


KCLBOT Designs 8 Mobile Robot Motion Controller Mount

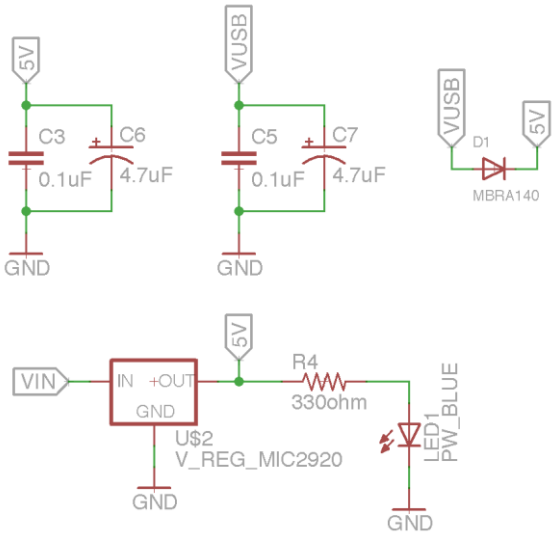
12.2 KCLBOT Electrical System



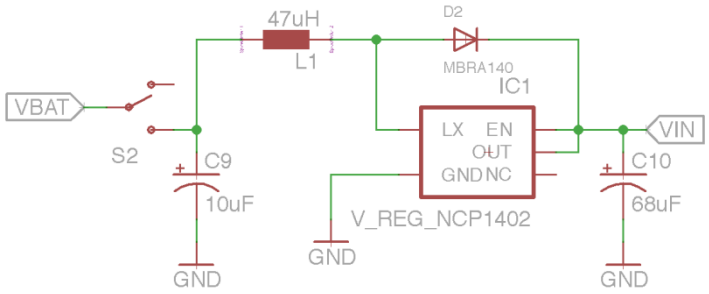
KCLBOT Electrical System 1 The ATMega328 Micro-controller Setup



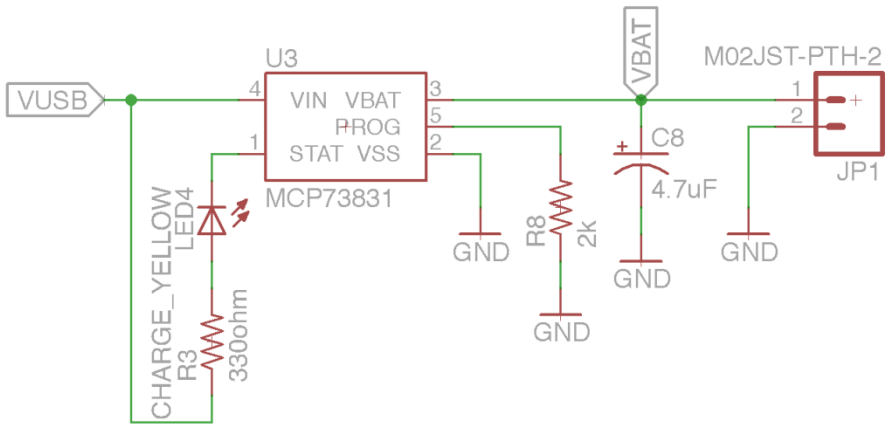
KCLBOT Electrical System 2 System Interface & Communication Setup



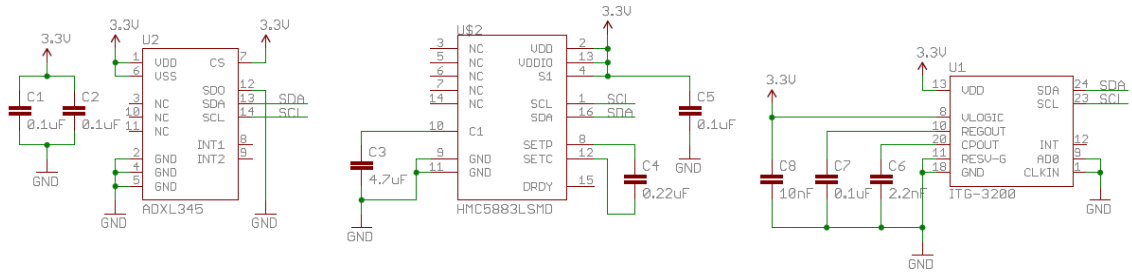
KCLBOT Electrical System 3 System Power Management & Protection



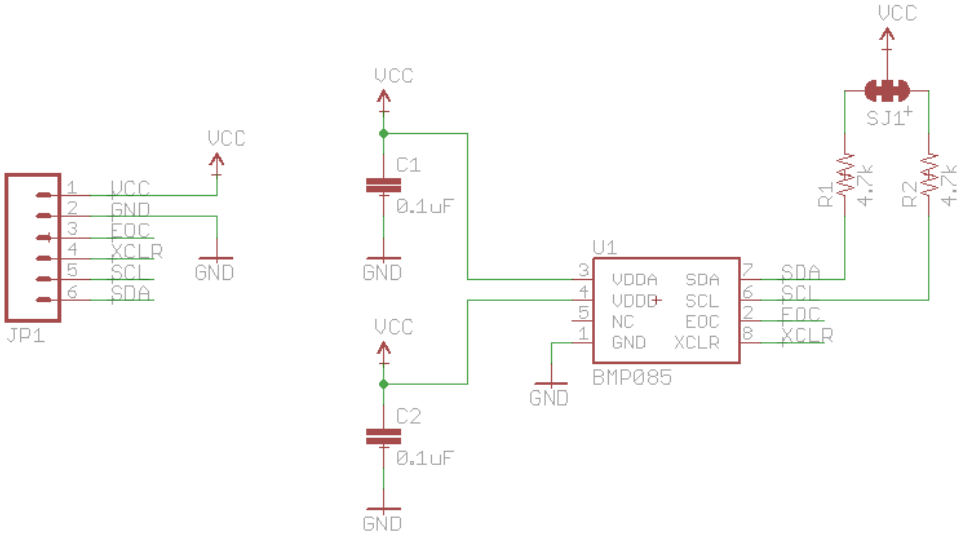
KCLBOT Electrical System 4 Power Step Boost DC-DC Setup



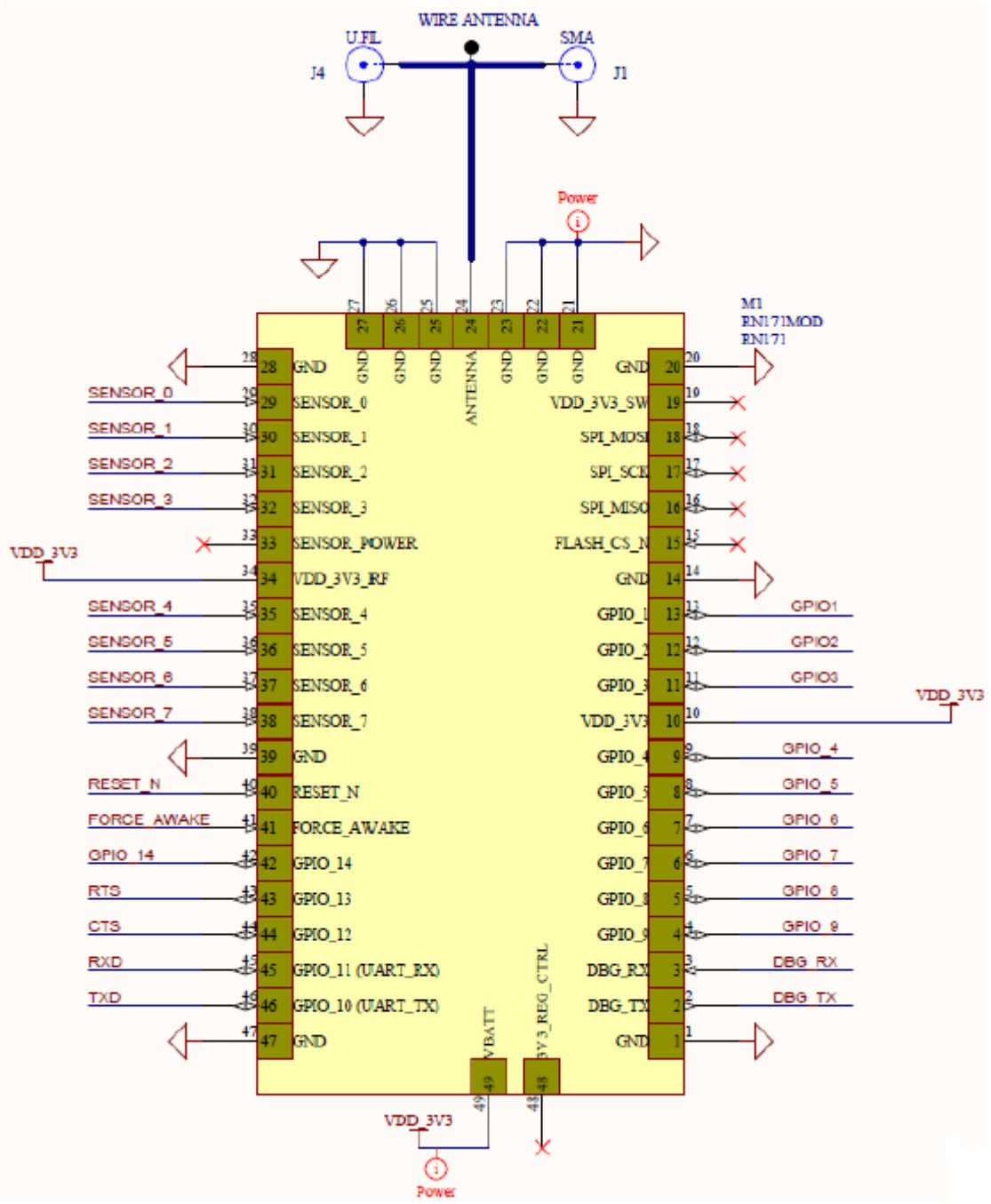
KCLBOT Electrical System 5 System LiPo Battery Charger



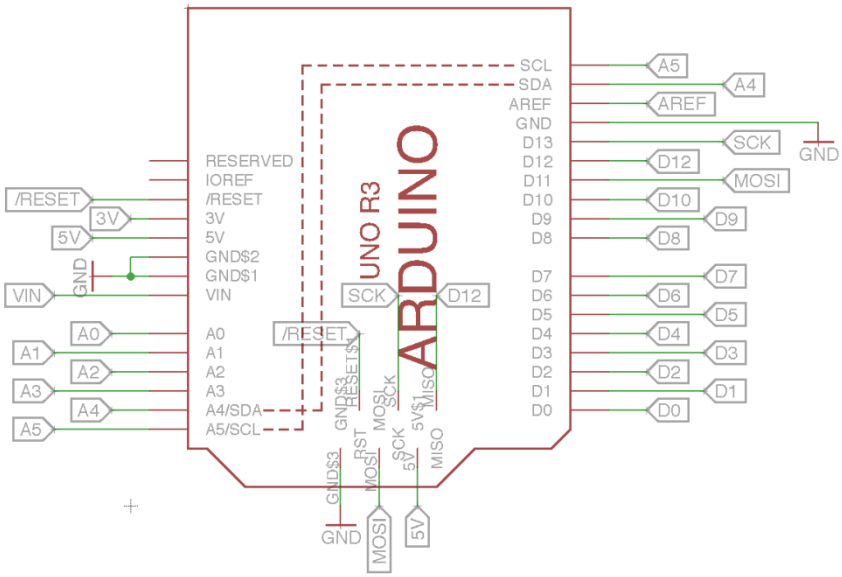
KCLBOT Electrical System 6 Inertial Measurement Unit (IMU) Schematic



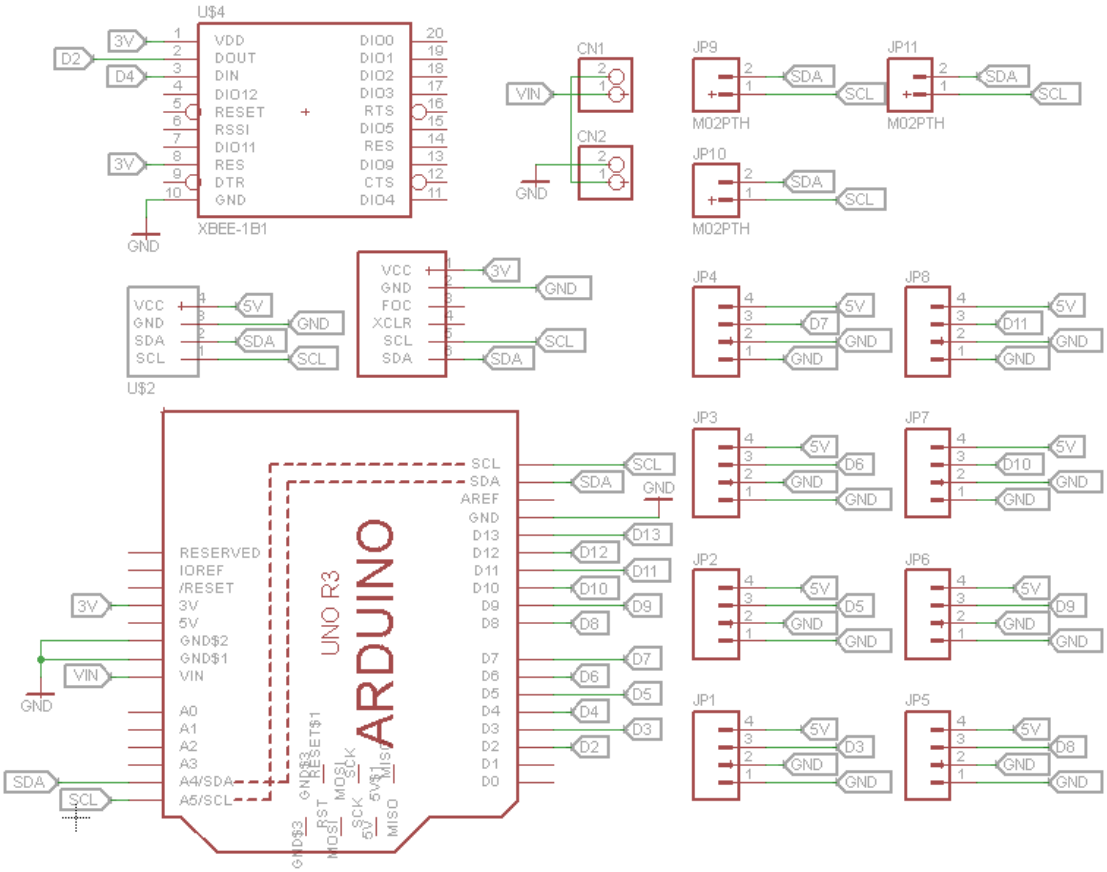
KCLBOT Electrical System 7 Digital Pressure Sensor Schematic



KCLBOT Electrical System 8 Serial Wireless System Schematic

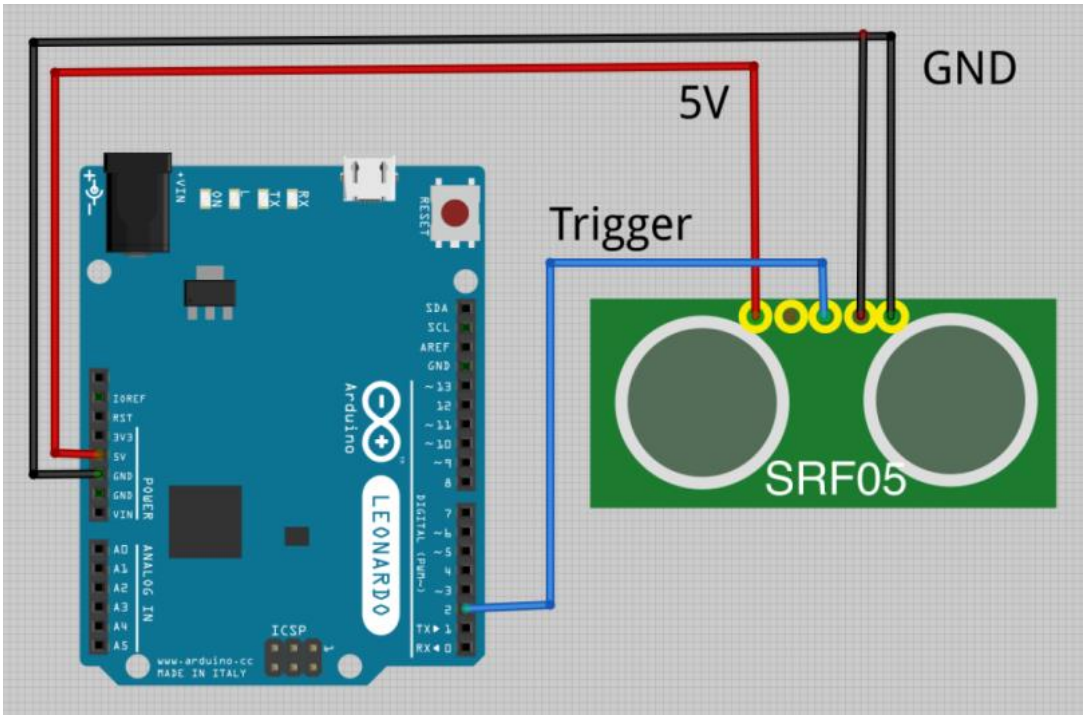


KCLBOT Electrical System 9 Arduino UNO R3 Setup

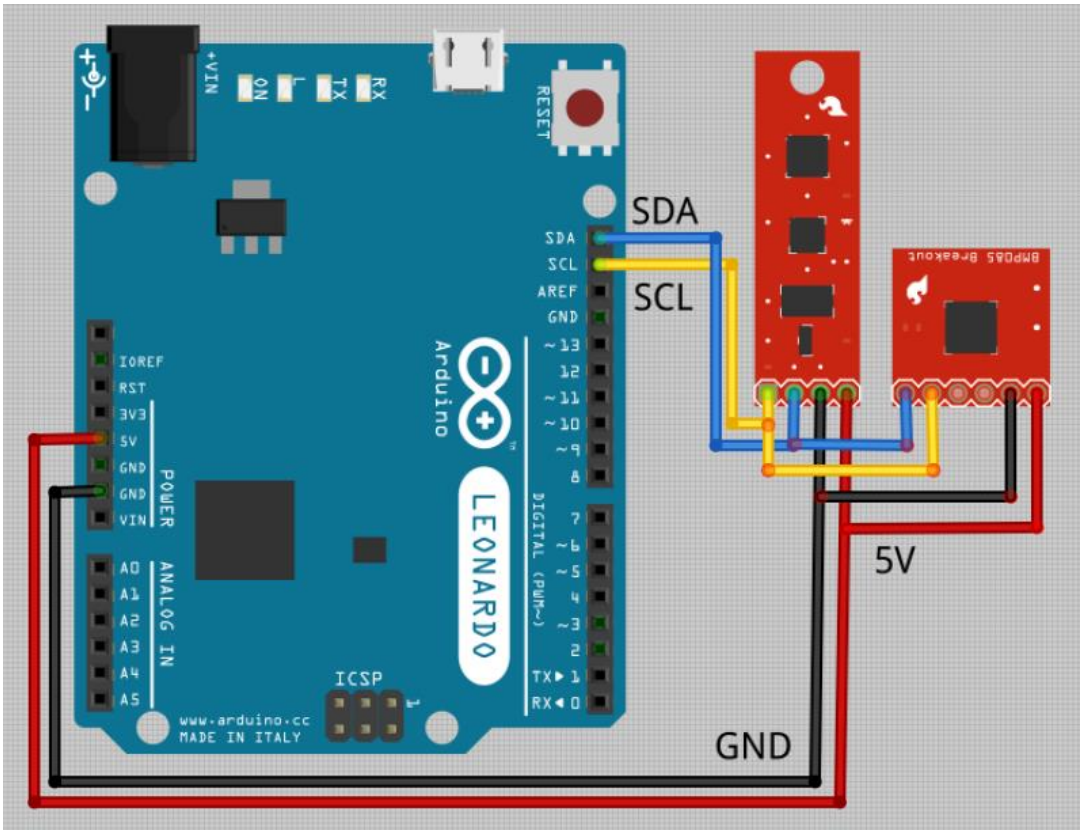


KCLBOT Electrical System 10 Arduino Sensor Shield

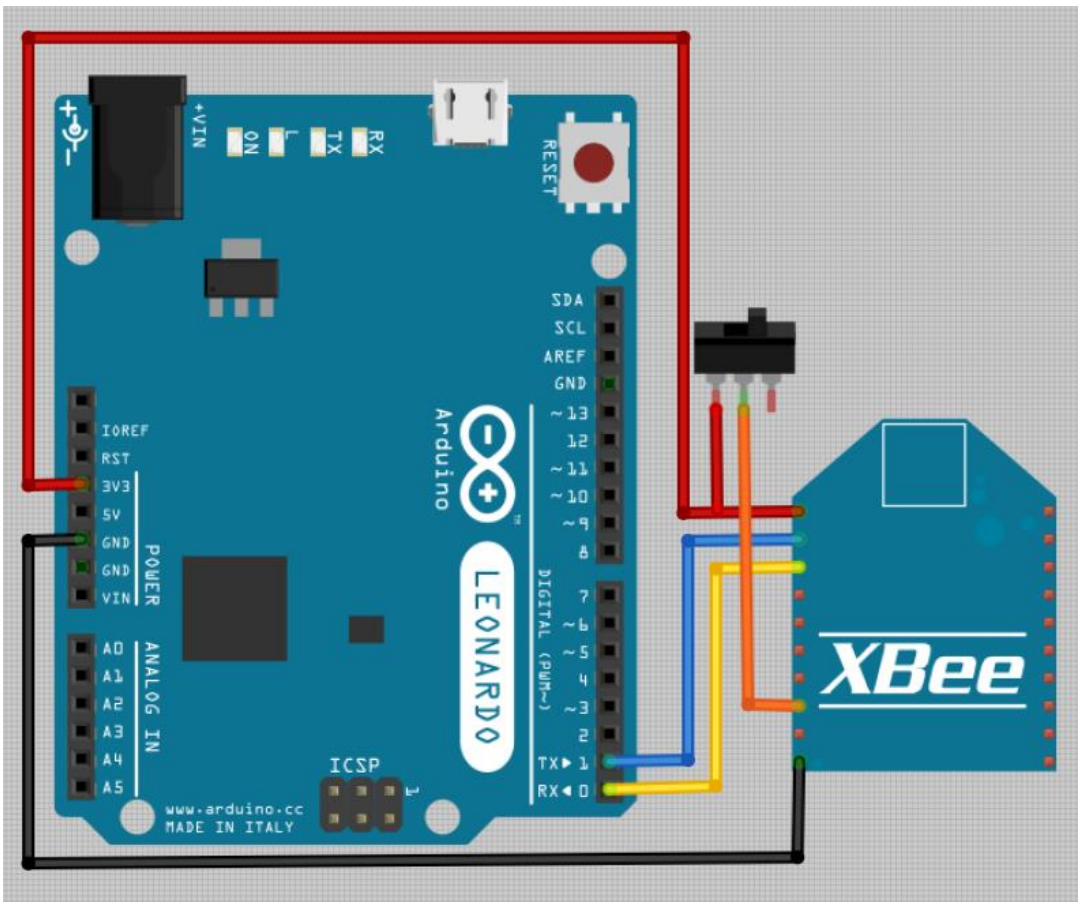
12.3 KCLBOT Arduino Configuration



KCLBOT Arduino Configuration 1 SRF05 Range Sensor Interface with Arduino



KCLBOT Arduino Configuration 2 9DOF IMU & BMP085 Interface with Arduino



KCLBOT Arduino Configuration 3 RN-171g Wireless Module Interface with Arduino

12.4 KCLBOT Arduino Software

```
int duration; //Duration of the pulse
int distance; //Distance variable
int TriggerPin = 2; //Configured SRF05SRF05 Trigger Pin

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  pinMode(TriggerPin, OUTPUT);
  digitalWrite(TriggerPin, LOW); //Set pin to low before sending a short high
to trigger ranging
  delayMicroseconds(2);
  digitalWrite(TriggerPin, HIGH); //Send a short 10 microsecond high burst on
pin to start ranging
  delayMicroseconds(10);
  digitalWrite(TriggerPin, LOW); //Set pin low again before waiting for pulse
to return
  pinMode(TriggerPin, INPUT);
  duration = pulseIn(TriggerPin, HIGH); //Reads echo pulse in from SRF05 in
micro seconds
  distance = duration/58;  //Dividing this by 58 gives us a distance in cm
  Serial.println(distance); //Print output result
  delay(100);
}
```

Arduino Source Code 1 SRF05 Range Sensor Interface with Arduino

```
#include <Wire.h>

#define BMP085_ADDRESS 0x77  // I2C address of BMP085

const unsigned char OSS = 0;  // Oversampling Setting

// Calibration values
int ac1;
int ac2;
int ac3;
unsigned int ac4;
unsigned int ac5;
unsigned int ac6;
int b1;
int b2;
int mb;
int mc;
int md;

long b5;

short temperature;
long pressure;

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  bmp085Calibration();
}

// Stores all of the bmp085's calibration values into global variables
// Calibration values are required to calculate temp and pressure
// This function should be called at the beginning of the program
void bmp085Calibration()
{
  ac1 = bmp085ReadInt(0xAA);
  ac2 = bmp085ReadInt(0xAC);
  ac3 = bmp085ReadInt(0xAE);
  ac4 = bmp085ReadInt(0xB0);
  ac5 = bmp085ReadInt(0xB2);
  ac6 = bmp085ReadInt(0xB4);
  b1 = bmp085ReadInt(0xB6);
  b2 = bmp085ReadInt(0xB8);
  mb = bmp085ReadInt(0xBA);
  mc = bmp085ReadInt(0xBC);
  md = bmp085ReadInt(0xBE);
}
```

Arduino Source Code 2 BMP085 Pressure Sensor Configuration

```
// Calculate temperature given ut.
// Value returned will be in units of 0.1 deg C
short bmp085GetTemperature(unsigned int ut)
{
  long x1, x2;

  x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
  x2 = ((long)mc << 11)/(x1 + md);
  b5 = x1 + x2;

  return ((b5 + 8)>>4);
}

// Calculate pressure given up
// calibration values must be known
// b5 is also required so bmp085GetTemperature(...) must be called first.
// Value returned will be pressure in units of Pa.
long bmp085GetPressure(unsigned long up)
{
  long x1, x2, x3, b3, b6, p;
  unsigned long b4, b7;

  b6 = b5 - 4000;
  // Calculate B3
  x1 = (b2 * (b6 * b6)>>12)>>11;
  x2 = (ac2 * b6)>>11;
  x3 = x1 + x2;
  b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;

  // Calculate B4
  x1 = (ac3 * b6)>>13;
  x2 = (b1 * ((b6 * b6)>>12))>>16;
  x3 = ((x1 + x2) + 2)>>2;
  b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;

  b7 = ((unsigned long)(up - b3) * (50000>>OSS));
  if (b7 < 0x80000000)
    p = (b7<<1)/b4;
  else
    p = (b7/b4)<<1;

  x1 = (p>>8) * (p>>8);
  x1 = (x1 * 3038)>>16;
  x2 = (-7357 * p)>>16;
  p += (x1 + x2 + 3791)>>4;

  return p;
}
```

Arduino Source Code 3 BMP085 Temperature & Pressure Functions

```
// Read 1 byte from the BMP085 at 'address'
char bmp085Read(unsigned char address)
{
  unsigned char data;

  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.send(address);
  Wire.endTransmission();

  Wire.requestFrom(BMP085_ADDRESS, 1);
  while(!Wire.available())
    ;

  return Wire.receive();
}

// Read 2 bytes from the BMP085
// First byte will be from 'address'
// Second byte will be from 'address'+1
int bmp085ReadInt(unsigned char address)
{
  unsigned char msb, lsb;

  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.send(address);
  Wire.endTransmission();

  Wire.requestFrom(BMP085_ADDRESS, 2);
  while(Wire.available()<2)
    ;
  msb = Wire.receive();
  lsb = Wire.receive();

  return (int) msb<<8 | lsb;
}

// Read the uncompensated temperature value
unsigned int bmp085ReadUT()
{
  unsigned int ut;

  // Write 0x2E into Register 0xF4
  // This requests a temperature reading
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.send(0xF4);
  Wire.send(0x2E);
  Wire.endTransmission();

  // Wait at least 4.5ms
  delay(5);

  // Read two bytes from registers 0xF6 and 0xF7
  ut = bmp085ReadInt(0xF6);
  return ut;
}
```

Arduino Source Code 4 BMP085 Helper Functions Part1

```
// Read the uncompensated pressure value
unsigned long bmp085ReadUP()
{
  unsigned char msb, lsb, xlsb;
  unsigned long up = 0;

  // Write 0x34+(OSS<<6) into register 0xF4
  // Request a pressure reading w/ oversampling setting
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.send(0xF4);
  Wire.send(0x34 + (OSS<<6));
  Wire.endTransmission();

  // Wait for conversion, delay time dependent on OSS
  delay(2 + (3<<OSS));

  // Read register 0xF6 (MSB), 0xF7 (LSB), and 0xF8 (XLSB)
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.send(0xF6);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 3);

  // Wait for data to become available
  while(Wire.available() < 3)
    ;
  msb = Wire.receive();
  lsb = Wire.receive();
  xlsb = Wire.receive();

  up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) | (unsigned
long) xlsb) >> (8-OSS);

  return up;
}
```

Arduino Source Code 5 BMP085 Helper Functions Part 2

```
void loop()
{
  temperature = bmp085GetTemperature(bmp085ReadUT());
  pressure = bmp085GetPressure(bmp085ReadUP());
  Serial.print("Temperature: ");
  Serial.print(temperature, DEC);
  Serial.println(" *0.1 deg C");
  Serial.print("Pressure: ");
  Serial.print(pressure, DEC);
  Serial.println(" Pa");
  Serial.println();
  delay(1000);
}
```

Arduino Source Code 6 BMP085 Main Loop

```
// SENSOR CALIBRATION
// Accelerometer
// "accel x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX  Z_MIN/Z_MAX"
#define ACCEL_X_MIN ((float) -250)
#define ACCEL_X_MAX ((float) 250)
#define ACCEL_Y_MIN ((float) -250)
#define ACCEL_Y_MAX ((float) 250)
#define ACCEL_Z_MIN ((float) -250)
#define ACCEL_Z_MAX ((float) 250)

// Magnetometer (standard calibration mode)
// "magn x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX  Z_MIN/Z_MAX"
#define MAGN_X_MIN ((float) -600)
#define MAGN_X_MAX ((float) 600)
#define MAGN Y MIN ((float) -600)
#define MAGN_Y_MAX ((float) 600)
#define MAGN_Z_MIN ((float) -600)
#define MAGN_Z_MAX ((float) 600)

// Magnetometer (extended calibration mode)
//#define CALIBRATION__MAGN_USE_EXTENDED true
const float magn_ellipsoid_center[3] = {91.5, -13.5, -48.1};
const float magn_ellipsoid_transform[3][3] = {{0.902, -0.00354, 0.000636}, {-
0.00354, 0.9, -0.00599}, {0.000636, -0.00599, 1}};

// Gyroscope
// "gyro x,y,z (current/average) = .../OFFSET_X  .../OFFSET_Y  .../OFFSET_Z
#define GYRO_AVERAGE_OFFSET_X ((float) -42.05)
#define GYRO_AVERAGE_OFFSET_Y ((float) 96.20)
#define GYRO AVERAGE OFFSET Z ((float) -18.36)
```

Arduino Source Code 7 9DOF Sensor Calibration

```
// Sensor calibration scale and offset values
#define ACCEL_X_OFFSET ((ACCEL_X_MIN + ACCEL_X_MAX) / 2.0f)
#define ACCEL_Y_OFFSET ((ACCEL_Y_MIN + ACCEL_Y_MAX) / 2.0f)
#define ACCEL_Z_OFFSET ((ACCEL_Z_MIN + ACCEL_Z_MAX) / 2.0f)
#define ACCEL_X_SCALE (GRAVITY / (ACCEL_X_MAX - ACCEL_X_OFFSET))
#define ACCEL_Y_SCALE (GRAVITY / (ACCEL_Y_MAX - ACCEL_Y_OFFSET))
#define ACCEL_Z_SCALE (GRAVITY / (ACCEL_Z_MAX - ACCEL_Z_OFFSET))

#define MAGN_X_OFFSET ((MAGN_X_MIN + MAGN_X_MAX) / 2.0f)
#define MAGN_Y_OFFSET ((MAGN_Y_MIN + MAGN_Y_MAX) / 2.0f)
#define MAGN_Z_OFFSET ((MAGN_Z_MIN + MAGN_Z_MAX) / 2.0f)
#define MAGN_X_SCALE (100.0f / (MAGN_X_MAX - MAGN_X_OFFSET))
#define MAGN_Y_SCALE (100.0f / (MAGN_Y_MAX - MAGN_Y_OFFSET))
#define MAGN_Z_SCALE (100.0f / (MAGN_Z_MAX - MAGN_Z_OFFSET))

// Gain for gyroscope (ITG-3200)
#define GYRO_GAIN 0.06957 // Same gain on all axes
#define GYRO_SCALED_RAD(x) (x * TO_RAD(GYRO_GAIN)) // Calculate the scaled
gyro readings in radians per second
```

Arduino Source Code 8 9DOF Sensor Calibration Scale & Offset

```
// DCM parameters
#define Kp_ROLLPITCH 0.02f
#define Ki_ROLLPITCH 0.00002f
#define Kp_YAW 1.2f
#define Ki_YAW 0.00002f

#define STATUS_LED_PIN 13  // Pin number of status LED
#define GRAVITY 256.0f // "1G reference" used for DCM filter and
accelerometer calibration
#define TO_RAD(x) (x * 0.01745329252)  // *pi/180
#define TO_DEG(x) (x * 57.2957795131)  // *180/pi

// DCM variables
float MAG_Heading;
float Accel_Vector[3]= {0, 0, 0}; // Store the acceleration in a vector
float Gyro_Vector[3]= {0, 0, 0}; // Store the gyros turn rate in a vector
float Omega_Vector[3]= {0, 0, 0}; // Corrected Gyro_Vector data
float Omega_P[3]= {0, 0, 0}; // Omega Proportional correction
float Omega_I[3]= {0, 0, 0}; // Omega Integrator
float Omega[3]= {0, 0, 0};
float errorRollPitch[3] = {0, 0, 0};
float errorYaw[3] = {0, 0, 0};
float DCM_Matrix[3][3] = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
float Update_Matrix[3][3] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
float Temporary_Matrix[3][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
```

Arduino Source Code 9 9DOF Sensor DCM Parameters & Variables

```
// Sensor variables
float accel[3];  // Actually stores the NEGATED acceleration (equals gravity,
if board not moving).
float accel_min[3];
float accel_max[3];

float magnetom[3];
float magnetom_min[3];
float magnetom_max[3];
float magnetom_tmp[3];

float gyro[3];
float gyro_average[3];
int gyro_num_samples = 0;

// Euler angles
float yaw;
float pitch;
float roll;

// DCM timing in the main loop
unsigned long timestamp;
unsigned long timestamp_old;
float G_Dt; // Integration time for DCM algorithm

// Output-state variables
boolean output_stream_on;
boolean output_single_on;
int curr_calibration_sensor = 0;
boolean reset_calibration_session_flag = true;
int num_accel_errors = 0;
int num_magn_errors = 0;
int num_gyro_errors = 0;
```

Arduino Source Code 10 9DOF Sensor Setup Variables

```
// Apply calibration to raw sensor readings
void compensate_sensor_errors() {
    // Compensate accelerometer error
    accel[0] = (accel[0] - ACCEL_X_OFFSET) * ACCEL_X_SCALE;
    accel[1] = (accel[1] - ACCEL_Y_OFFSET) * ACCEL_Y_SCALE;
    accel[2] = (accel[2] - ACCEL_Z_OFFSET) * ACCEL_Z_SCALE;

    // Compensate magnetometer error
    for (int i = 0; i < 3; i++)
      magnetom_tmp[i] = magnetom[i] - magn_ellipsoid_center[i];
    Matrix_Vector_Multiply(magn_ellipsoid_transform, magnetom_tmp, magnetom);

    // Compensate gyroscope error
    gyro[0] -= GYRO_AVERAGE_OFFSET_X;
    gyro[1] -= GYRO_AVERAGE_OFFSET_Y;
    gyro[2] -= GYRO_AVERAGE_OFFSET_Z;
}
```

Arduino Source Code 11 9DOF Sensor Calibrate Raw Sensor Variables

```
void read_sensors() {
  Read_Gyro(); // Read gyroscope
  Read_Accel(); // Read accelerometer
  Read_Magn(); // Read magnetometer
}

// Read every sensor and record a time stamp
// Init DCM with unfiltered orientation
void reset_sensor_fusion() {
  float temp1[3];
  float temp2[3];
  float xAxis[] = {1.0f, 0.0f, 0.0f};

  read_sensors();
  timestamp = millis();

  // GET PITCH
  // Using y-z-plane-component/x-component of gravity vector
  pitch = -atan2(accel[0], sqrt(accel[1] * accel[1] + accel[2] * accel[2]));

  // GET ROLL
  // Compensate pitch of gravity vector
  Vector_Cross_Product(temp1, accel, xAxis);
  Vector_Cross_Product(temp2, xAxis, temp1);
  roll = atan2(temp2[1], temp2[2]);

  // GET YAW
  Compass_Heading();
  yaw = MAG_Heading;

  // Init rotation matrix
  init_rotation_matrix(DCM_Matrix, yaw, pitch, roll);
}
```

Arduino Source Code 12 9DOF Sensor Read Sensors & Reset Sensor Fusion Functions

```
void setup()
{
  // Init serial output
  Serial.begin(OUTPUT__BAUD_RATE);

  // Init status LED
  pinMode (STATUS_LED_PIN, OUTPUT);
  digitalWrite(STATUS_LED_PIN, LOW);

  // Init sensors
  delay(50);  // Give sensors enough time to start
  I2C_Init();
  Accel_Init();
  Magn_Init();
  Gyro_Init();

  // Read sensors, init DCM algorithm
  delay(20);  // Give sensors enough time to collect data
  reset_sensor_fusion();
}
```

Arduino Source Code 13 9DOF Sensor Main Setup

```
void loop()
{
  if((millis() - timestamp) >= OUTPUT__DATA_INTERVAL)
  {
    timestamp_old = timestamp;
    timestamp = millis();
    if (timestamp > timestamp_old)
      G_Dt = (float) (timestamp - timestamp_old) / 1000.0f;
    else G_Dt = 0;

    // Update sensor readings
    read_sensors();

    // Apply sensor calibration
    compensate_sensor_errors();

    // Run DCM algorithm
    Compass_Heading(); // Calculate magnetic heading
    Matrix_update();
    Normalize();
    Drift_correction();
    Euler_angles();
  }
}
```

Arduino Source Code 14 9DOF Sensor Main Loop

```
void Compass_Heading()
{
  float mag_x;
  float mag_y;
  float cos_roll;
  float sin_roll;
  float cos_pitch;
  float sin_pitch;

  cos_roll = cos(roll);
  sin_roll = sin(roll);
  cos_pitch = cos(pitch);
  sin_pitch = sin(pitch);

  // Tilt compensated magnetic field X
  mag_x = magnetom[0] * cos_pitch + magnetom[1] * sin_roll * sin_pitch +
magnetom[2] * cos_roll * sin_pitch;
  // Tilt compensated magnetic field Y
  mag_y = magnetom[1] * cos_roll - magnetom[2] * sin_roll;
  // Magnetic Heading
  MAG_Heading = atan2(-mag_y, mag_x);
}
```

Arduino Source Code 15 9DOF Sensor Compass Heading Function

```
void Normalize(void)
{
  float error=0;
  float temporary[3][3];
  float renorm=0;

  error= -Vector_Dot_Product(&DCM_Matrix[0][0],&DCM_Matrix[1][0])*.5;

  Vector_Scale(&temporary[0][0], &DCM_Matrix[1][0], error);
  Vector_Scale(&temporary[1][0], &DCM_Matrix[0][0], error);

  Vector_Add(&temporary[0][0], &temporary[0][0], &DCM_Matrix[0][0]);
  Vector_Add(&temporary[1][0], &temporary[1][0], &DCM_Matrix[1][0]);

  Vector_Cross_Product(&temporary[2][0],&temporary[0][0],&temporary[1][0]);

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[0][0],&temporary[0][0]));
  Vector_Scale(&DCM_Matrix[0][0], &temporary[0][0], renorm);

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[1][0],&temporary[1][0]));
  Vector_Scale(&DCM_Matrix[1][0], &temporary[1][0], renorm);

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[2][0],&temporary[2][0]));
  Vector_Scale(&DCM_Matrix[2][0], &temporary[2][0], renorm);
}
```

Arduino Source Code 16 9DOF Sensor DCM Normalization Function

```
void Drift_correction(void)
{
  float mag_heading_x;
  float mag_heading_y;
  float errorCourse;
  static float Scaled_Omega_P[3];
  static float Scaled_Omega_I[3];
  float Accel_magnitude;
  float Accel_weight;

  //Roll and Pitch
  // Calculate the magnitude of the accelerometer vector
  Accel_magnitude = sqrt(Accel_Vector[0]*Accel_Vector[0] +
Accel_Vector[1]*Accel_Vector[1] + Accel_Vector[2]*Accel_Vector[2]);
  Accel_magnitude = Accel_magnitude / GRAVITY; // Scale to gravity.

  // Weight for accelerometer info (<0.5G = 0.0, 1G = 1.0 , >1.5G = 0.0)
  Accel_weight = constrain(1 - 2*abs(1 - Accel_magnitude),0,1);  //


Vector_Cross_Product(&errorRollPitch[0],&Accel_Vector[0],&DCM_Matrix[2][0]);
  Vector_Scale(&Omega_P[0],&errorRollPitch[0],Kp_ROLLPITCH*Accel_weight);


Vector_Scale(&Scaled_Omega_I[0],&errorRollPitch[0],Ki_ROLLPITCH*Accel_weight)
;
  Vector_Add(Omega_I,Omega_I,Scaled_Omega_I);

  //Yaw
  mag_heading_x = cos(MAG_Heading);
  mag_heading_y = sin(MAG_Heading);
  errorCourse=(DCM_Matrix[0][0]*mag_heading_y) -
(DCM_Matrix[1][0]*mag_heading_x);
  Vector_Scale(errorYaw,&DCM_Matrix[2][0],errorCourse);

  Vector_Scale(&Scaled_Omega_P[0],&errorYaw[0],Kp_YAW);
  Vector_Add(Omega_P,Omega_P,Scaled_Omega_P);

  Vector_Scale(&Scaled_Omega_I[0],&errorYaw[0],Ki_YAW);
  Vector_Add(Omega_I,Omega_I,Scaled_Omega_I);
}
```

Arduino Source Code 17 9DOF Sensor DCM Drift Compensation Function

```
void Matrix_update(void)
{
  Gyro_Vector[0]=GYRO_SCALED_RAD(gyro[0]); //gyro x roll
  Gyro_Vector[1]=GYRO_SCALED_RAD(gyro[1]); //gyro y pitch
  Gyro_Vector[2]=GYRO_SCALED_RAD(gyro[2]); //gyro z yaw

  Accel_Vector[0]=accel[0];
  Accel_Vector[1]=accel[1];
  Accel_Vector[2]=accel[2];

  Vector_Add(&Omega[0], &Gyro_Vector[0], &Omega_I[0]);  //adding proportional
term
  Vector_Add(&Omega_Vector[0], &Omega[0], &Omega_P[0]); //adding Integrator
term

  Update_Matrix[0][0]=0;
  Update_Matrix[0][1]=-G_Dt*Omega_Vector[2];//-z
  Update_Matrix[0][2]=G_Dt*Omega_Vector[1];//y
  Update_Matrix[1][0]=G_Dt*Omega_Vector[2];//z
  Update_Matrix[1][1]=0;
  Update_Matrix[1][2]=-G_Dt*Omega_Vector[0];//-x
  Update_Matrix[2][0]=-G_Dt*Omega_Vector[1];//-y
  Update_Matrix[2][1]=G_Dt*Omega_Vector[0];//x
  Update_Matrix[2][2]=0;

  Matrix_Multiply(DCM_Matrix,Update_Matrix,Temporary_Matrix);

  for(int x=0; x<3; x++) //Matrix Addition (update)
  {
    for(int y=0; y<3; y++)
    {
      DCM_Matrix[x][y]+=Temporary_Matrix[x][y];
    }
  }
}
```

Arduino Source Code 18 9DOF Sensor DCM Matrix Update

```
//The dot product of two vectors
float Vector_Dot_Product(const float v1[3], const float v2[3])
{
  float result = 0;

  for(int c = 0; c < 3; c++)
  {
    result += v1[c] * v2[c];
  }

  return result;
}

//The cross product of two vectors
void Vector_Cross_Product(float out[3], const float v1[3], const float v2[3])
{
  out[0] = (v1[1] * v2[2]) - (v1[2] * v2[1]);
  out[1] = (v1[2] * v2[0]) - (v1[0] * v2[2]);
  out[2] = (v1[0] * v2[1]) - (v1[1] * v2[0]);
}

//Multiply the vector by a scalar
void Vector_Scale(float out[3], const float v[3], float scale)
{
  for(int c = 0; c < 3; c++)
  {
    out[c] = v[c] * scale;
  }
}

//Adds two vectors
void Vector_Add(float out[3], const float v1[3], const float v2[3])
{
  for(int c = 0; c < 3; c++)
  {
    out[c] = v1[c] + v2[c];
  }
}
```

Arduino Source Code 19 9DOF Sensor Matrix Operations

```
// Multiply two 3x3 matrices
void Matrix_Multiply(const float a[3][3], const float b[3][3], float
out[3][3])
{
  for(int x = 0; x < 3; x++)  // rows
  {
    for(int y = 0; y < 3; y++)  // columns
    {
      out[x][y] = a[x][0] * b[0][y] + a[x][1] * b[1][y] + a[x][2] * b[2][y];
    }
  }
}

// Multiply 3x3 matrix with vector
void Matrix_Vector_Multiply(const float a[3][3], const float b[3], float
out[3])
{
  for(int x = 0; x < 3; x++)
  {
    out[x] = a[x][0] * b[0] + a[x][1] * b[1] + a[x][2] * b[2];
  }
}

// Initialize the rotation matrix using Euler angles
void init_rotation_matrix(float m[3][3], float yaw, float pitch, float roll)
{
  float c1 = cos(roll);
  float s1 = sin(roll);
  float c2 = cos(pitch);
  float s2 = sin(pitch);
  float c3 = cos(yaw);
  float s3 = sin(yaw);

  m[0][0] = c2 * c3;
  m[0][1] = c3 * s1 * s2 - c1 * s3;
  m[0][2] = s1 * s3 + c1 * c3 * s2;

  m[1][0] = c2 * s3;
  m[1][1] = c1 * c3 + s1 * s2 * s3;
  m[1][2] = c1 * s2 * s3 - c3 * s1;

  m[2][0] = -s2;
  m[2][1] = c2 * s1;
  m[2][2] = c1 * c2;
}
```

Arduino Source Code 20 9DOF Sensor Matrix Operation Functions

12.5 KCLBOT Firmware & Software

The KCLBOT is built around Arduino hardware and, as such, is programmed with the Arduino integrated development environment (IDE). The Arduino IDE is a cross platform application that is written in Java, and was developed from Processing programming language and the Wiring project. Arduino software is developed using C or C++ and the compiler then turns this into machine code and pushes it onto the microcontroller. The Arduino IDE uses GNU toolchain and the AVR Libc to compile the firmware applications and typically uses avrdude to upload the applications to the microcontroller.

The Arduino software structure is dependent on two core functions, setup() which is a function that is run only once at the start of the application and is often used to initialize settings, and loop() which, like its name, is a continuously repeating function that runs until the microcontroller is powered off.

```
#include <SoftwareSerial.h>
SoftwareSerial wifiSerial(13,12);

void setup()
{
  Serial.begin(115200);
  wifiSerial.begin(57600);
}
```

The setup code presented above shows how the wireless module, which is connected via the serial pins 12 and 13, is configured at a baud rate of 57600bps using the software serial drivers. The Arduino built serial port for debugging is also configured at a baud rate of 115200bps. The wireless module performed better at the lower baud rate with no bit errors, but at the higher 115200bps baud rate, the module returned bit errors.

```
void motorController(int leftMotorDirection, int leftMotorSpeed, int
rightMotorDirection, int rightMotorSpeed)
{
  if (leftMotorDirection == 0)
  {
    analogWrite(leftMotorPinOne, HIGH);
    analogWrite(leftMotorPinTwo, leftMotorSpeed);
  }
  else if (leftMotorDirection == 1)
  {
    analogWrite(leftMotorPinOne, leftMotorSpeed);
    analogWrite(leftMotorPinTwo, HIGH);
  }

  if (rightMotorDirection == 0)
  {
    analogWrite(rightMotorPinOne, HIGH);
    analogWrite(rightMotorPinTwo, rightMotorSpeed);
  }
  else if (rightMotorDirection == 1)
  {
    analogWrite(rightMotorPinOne, rightMotorSpeed);
    analogWrite(rightMotorPinTwo, HIGH);
  }
}
```

The motors are controlled using the function presented above. The function requires four inputs, the direction of motion for each motor accepting a binary input, and the speed of the motors accepting a value between 0 and 256.

```
void loop()
{
    byte data[5];

    while(wifiSerial.available()) {
        int bytesAvailable = wifiSerial.available();

        if (bytesAvailable == 5)
        { //process motor command }
        else if (bytesAvailable == 2)
        { //send analog PIN data }
        else if (bytesAvailable == 3)
        { //set digital PIN state }
        else if (bytesAvailable == 1)
        { //send all analog PIN values }
        else
        { //dispose of erroneous values }
    }

    if (Serial.available()) {
        wifiSerial.write(Serial.read());
    }
}
```

The code above shows the data processing loop which waits for instructions from the wireless module before carrying out an action. The largest instruction is five bytes long, which belongs to

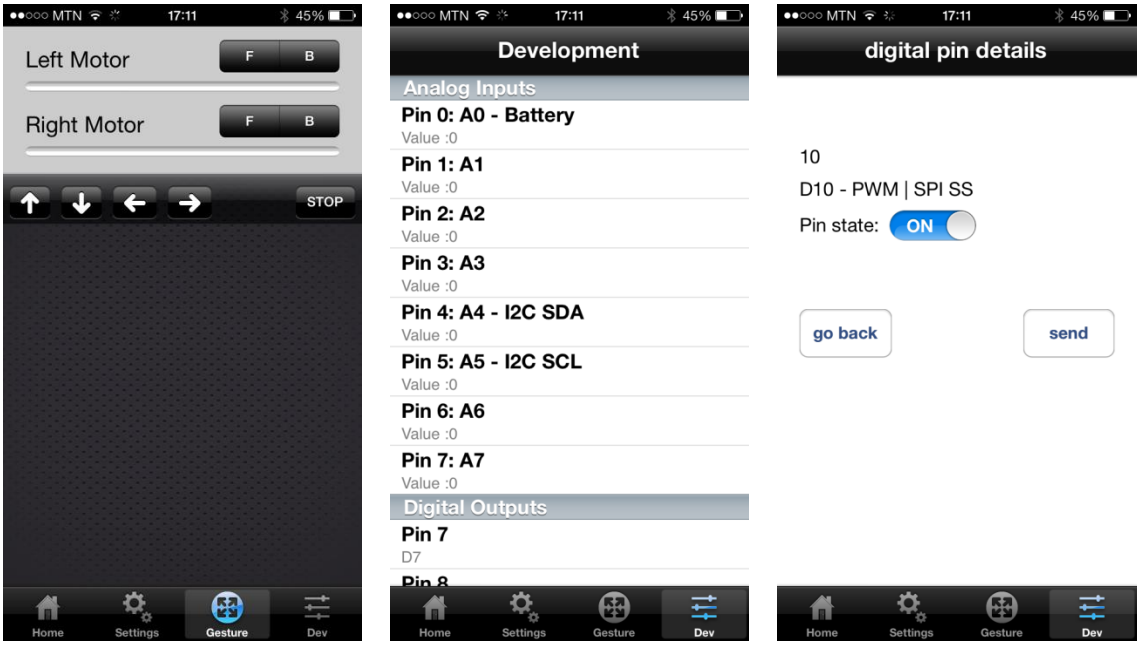
the motor controller. The first byte is a marker for the instruction and following data bytes are the values for the motor controller function. The two byte instruction sends the analog PIN value, the first byte is a marker for the instruction and the second byte is the PIN address for the request. The three byte instruction is used to control the state of the digital PIN, the first byte is the marker for the instruction and the second byte is the PIN address followed by the third byte which defines the PIN's state as being high or low. The single byte instruction, if it has the correct marker, is used to send all the available analog PIN values. Finally, for all undefined data requests, the process will dispose of the data.

```
if (bytesAvailable == 2){
    for (int i = 0; i < bytesAvailable; i++)
    { data[i] = wifiSerial.read(); }

    if ((int)data[0] == 2){
        Serial.println("RX - A");
        byte sendData[4];
        sendData[0] = 0x02;
        sendData[1] = data[1];

        int analogPinValue = analogRead((int)data[1]);
        sendData[2] = analogPinValue & 0xFF;
        sendData[3] = (analogPinValue >> 8) & 0xFF;

        wifiSerial.write(sendData, 4);
    }
}
```

The code here shows how a request for an analog PIN value is processed. The two byte value is received from the wireless module serial port, which indicates which PIN value is required. Once the request has been confirmed, the analog PIN value is read and a four byte data buffer is setup with the analog PIN marker, the analog PIN address, and the analog PIN value using two bytes for higher resolution. Finally, the four byte buffer values are written to wireless module for communication.

```
if (bytesAvailable == 3){
    for (int i = 0; i < bytesAvailable; i++)
        { data[i] = wifiSerial.read(); }

    if ((int)data[0] == 3){
        if ((int)data[2] == 0)
            { digitalWrite((int)data[1], LOW); }
        else if ((int)data[2] == 1)
            { digitalWrite((int)data[1], HIGH); }
    }
}
```

The code here describes how the microcontroller manages requests to change the state of the digital PINs. Once the three-byte-long data string is received, the second data byte identifies the PIN address and the third data byte defines the PIN state which is sent to the PIN address to make the PIN state high or low.



Motor Controller | Analog & Digital PIN Table | Digital PIN Controller

Figure 12-1 iOS Mobile Robot Software Interface

The graphical user interface (GUI), which is developed on the iOS platform, depicted in Figure 12-1, is composed of three major sections to control the mobile robot's motors, a screen to read the analog and digital PIN values, and a screen to control the digital PIN state.

```
- (void)sendUDPMotorControl:(int)leftDirection leftMotor:(int)leftMotor
rightDirection:(int)rightDirection rightMotor:(int)rightMotor
{
    Byte TxDataBytes[5];

    TxDataBytes[0] = 1;
    TxDataBytes[1] = leftDirection;
    TxDataBytes[2] = leftMotor;
    TxDataBytes[3] = rightDirection;
    TxDataBytes[4] = rightMotor;

    NSLog(@"d1: %d, d2: %d, d3: %d, d4: %d", TxDataBytes[1], TxDataBytes[2],
TxDataBytes[3], TxDataBytes[4]);

    NSData *TxData = [[NSData alloc] initWithBytes:TxDataBytes length:5];

    int portNumber = 0;
    NSString *ipAdrress = [[NSString alloc] init];
    for (Settings *set in arr) {
        ipAdrress = set.ipaddress;
        portNumber = [set.port intValue];
    }

    [udpSocket sendData:TxData toHost:ipAdrress port:portNumber withTimeout:-
1 tag:1];
}
```

The code here presents the function that is used to control the mobile robot's motors. The function collects four variables which are composed of the direction of the motion for the motors and the power value of each motor. The values are stored in a five-byte-long transmission buffer, the first value being the marker value for a motor control identifier; the four remaining values are motor values. After the values are stored in the buffer the values are communicated wirelessly using an asynchronous UDP communication protocol.

```
-(BOOL)onUdpSocket:(AsyncUdpSocket *)sock didReceiveData:(NSData *)data
           withTag:(long)tag
          fromHost:(NSString *)host
              port:(UInt16)port
{
    NSLog(@"Received UDP Packet");

    UInt8 *bytes = (UInt8 *)data.bytes;
    uint16_t updateValue = 0;

    for (int i = 1; i < 9; i++) {
        updateValue = (bytes[i * 2] << 8) | bytes[(i * 2) - 1];

        [analogPinValue replaceObjectAtIndex:(i - 1) withObject:[NSNumber
numberWithInt:updateValue]];
        [analogInputs setObject:analogPinValue forKey:@"PinNumber"];
    }

    float updateNewValue = 0;
    for (int i = 9; i < 12; i++) {
        updateNewValue = ((bytes[i * 2] << 8) | bytes[(i * 2) - 1]) - 360;

        [analogPinValue replaceObjectAtIndex:(i - 1) withObject:[NSNumber
numberWithInt:updateNewValue]];
        [analogInputs setObject:analogPinValue forKey:@"PinNumber"];
    }

    [self.tableViewUpdate reloadData];
    [udpSocket receiveWithTimeout:-1 tag:1];
    return YES;
}
```

The code presented here shows how the data from the microcontroller's analog PIN values are received from the UDP socket watcher. The iOS software is setup with a thread for the UDP socket watcher, which waits for data from the microcontroller. Once the data is confirmed as received, the function rebuilds the two-byte length values into a single float value, which is then updated on the user interface.

```
- (IBAction)sendPinState:(id)sender {
    int currentPinState = 0;

    if ([pinState isOn])
    { currentPinState = 1; }

    int pinData = [[digitalPinNumber objectAtIndex:indexNumberPushed]
integerValue];

    Byte TxDataBytes[3];

    TxDataBytes[0] = 3;
    TxDataBytes[1] = pinData;
    TxDataBytes[2] = currentPinState;

    NSData *TxData = [[NSData alloc] initWithBytes:TxDataBytes length:3];

    int portNumber = 0;
    NSString *ipAdrress = [[NSString alloc] init];
    for (Settings *set in arr) {
        ipAdrress = set.ipaddress;
        portNumber = [set.port intValue];
    }

    [udpSocket sendData:TxData toHost:ipAdrress port:portNumber withTimeout:-
1 tag:1];
}
```

Above is the code that communicates the state change for the digital pin states on the microcontroller. The data sent to the asynchronous UDP socket is three bytes long with the first byte identifying this digital state change request, the second identifying the digital PIN address and the third identifying the state the PIN should be changed to.